Evolution of Model-Based System Engineering Methodologies for the Design of Space Systems in the Advanced Stages of the Project

POLITECNICO DI TORINO

SCUOLA DI DOTTORATO

Dottorato in Ingegneria Aerospaziale XXVI Ciclo

Ph.D. Thesis

Evolution of Model-Based System Engineering Methodologies for the Design of Space Systems in the Advanced Stages of the Project (Phases B-C)

Academic Tutor:
Prof. Paolo Maggiore

Company Tutor:
Dott. Valter Basso
Ing. Mauro Pasquinelli

Student:
Michele Cencetti

December 2013

To my family

Contents

List of Figures

List of Tables

Abstract

The main topic of the present work is addressed to the evaluation of the possible improvements that can be achieved with the integration of Model Based System Engineering Methodologies in the advanced phases of space project. In particular a model based approach will be proposed for two main aspects directly affecting the design phases of complex systems. The first one is represented by the management of design options that becomes difficult to monitor as the project proceeds, increasing the amount of data to take into consideration. The other one is represented by the integration between Multidisciplinary Design Optimization (MDO) techniques and a Model Based System Engineering (MBSE) environment. The aim of the research activity concerns the feasibility of such connection in order to assess actual advantages and possible drawbacks. In this last case the objective is to show how the Multidisciplinary Design Optimization (MDO) methods may be managed in the context of a MBSE environment with respect to the traditional design approach. In particular this analysis is addressed to the demonstration of the benefits of MBSE methodology and MDO techniques considering a space system reference case. In the first part of the thesis a briefly description of the problem statement is introduced to better explain the subjects of the following chapters. In particular the reasons and the related purposes that have animated this work are considered. In the next section the state of the art about the considered approach is presented, providing a background for the following activities. In this context a wider analysis of the motivations and thesis objectives is considered. The following chapters deals with the survey and critical assessment of the main work related to this thesis. The analysis, design and implementation of the proposed framework are considered in the next sections. At the end of this part the results obtained are presented without arguing about the related benefits or drawbacks, which are considered in the following. A critical assessment of the results is then presented, analyzing the main contributions and related disadvantages with respect to the current approaches. In the next section the incoming activities and further developments are presented. The final part concerns at last the summary conclusions of the work done.

Acknowledgments

"I don't like honors. I'm appreciated for the work that I did, and for people who appreciate it, and I notice that other physicists use my work. I don't need anything else. I don't think there's any sense to anything else. I don't see that it makes any point that someone in the Swedish Academy decides that this work is noble enough to receive a prize. I've already got the prize. The prize is the pleasure of finding the thing out, the kick in the discovery, the observation that other people use it. Those are the real things. The honors are unreal to me. I don't believe in honors. It bothers me, honors. Honors is epilets, honors is uniforms"

Richard Phillips Feynman

With these few lines I thank all the people who were close to me, encouraged my efforts and supported my decisions. A special thanks goes also to Thales Alenia Space Italy for the opportunity, the advice and the expertise demonstrated during my PhD studentship.

Acronyms

AAO	All At Once
AFT	Architecture Framework Tool
AIT	Assembly Integration and Test
AMPL	A Mathematical Programming Language
ANN	Artificial Neural Network
AR	Acceptance Review
ASV	Active Set Vector
ATDD	Acceptance Test Driven Development
BDD	Block Definition Diagram
BIM	Building Information Model
BLISS	Bilevel Integrated System Synthesis
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CAM	Computer Aided Manufacturing
CAO	Computer Aided Optimization
CCB	Configuration Control Board
CDF	Concurrent Design Facility
CDR	Critical Design Review
CRR	Commissioning Result Review
CE	Concurrent Engineering
CER	Cost Estimating Relationship
CI	Continuous Integration
CM	Configuration Management
CNES	Centre National d'Etudes Spatiales
CO	Collaborative Optimization
COSE	COllaborative System Engineering

CPV	Common Pressure Vessel
CSA	Configuration Status Accounting
CWE	Collaborative Working Environment
DA	Discipline Analysis
DACE	Design/Analysis of Computer Experiments
DAKOTA	Design and Analysis toolKit for Optimization and Terascale Applications
DEVICE	Distributed Environment for Virtual Integrated Collaborative
DM	Data Management
DoD	Department of Defense
DOD	Depth Of Discharge
DOE	Design of Experiments
DOORS	Dynamic Object Oriented Requirements System
DSL	Domain Specific Language
DSM	Design Structure Matrix
DST	Domain Specific Tool
DSTE	Dempster-Shafer Theory of Evidence
DVV	Derivative Variables Vector
DXF	Drawing Interchange Format
DXL	DOORS eXtension Language
EA	Evolutionary Algorithms
ECF	Concurrent Engineering Facility
ECSS	European Cooperation for Space Standardization
EFFBD	Enhanced Functional Flow Block Diagram
EGO	Efficient Global Optimization
EIF	Efficient Improvement Function
ELR	End of Life Review
EPS	Electrical Power Subsystem
FAI	Field Aligned Plasma Irregularities
FCGI	Fast Common Getaway Interface
FDD	Feature Driven Development
FIDO	Framework for Interdisciplinary Design and Optimization

FMEA	Failure Mode and Effects Analysis
FMECA	Failure Mode Effects and Criticality Analysis
FRR	Flight Readiness Review
GA	Genetic Algorithm
GLOW	Gross Lift Off Weight
GP	Gaussian Process
GMM	Geometrical Mathematical Model
GSE	Ground Support Equipment
HLA	High Level Architecture
IBD	Internal Block Diagram
ICT	Information and Communication Technology
ICME	Integrated Model Centric Engineering
IDF	Individual Discipline Feasible
IDM	Integrated Design Model
IGES	Initial Graphics Exchange Specification
INCOSE	International Council on System Engineering
IPC	Inter Process Communication
IPG	Information Power Grids
IPV	Individual Pressure Vessel
ISO	International Standard Organization
ISR	Incoherent Scatter Radar
ISS	International Space Station
ITA	Ion Thruster Assembly
I&T	Integration and Test
IVP	Interval-valued Probability
JDBC	Java DataBase Connectivity
JEO	Jupiter Europa Orbiter
JNI	Java Native Interface
JSON	JavaScript Object Notation
KSA	Knowledge Skills Abilities
LHS	Latin Hypercube Sampling

LRR	Launch Readiness Review
MARS	Multivariate Adaptive Regression Splines
MBED	Model Based Engineering Design
MBSE	Model Based System Engineering
MCR	Mission Close-out Review
MDA	Model Driven Architecture
MDA	Multidisciplinary Design Analysis
MDAO	Multidisciplinary Design Analysis and Optimization
MDE	Model Driven Engineering
MDF	Multidisciplinary Feasible
MDO	Multidisciplinary Design Optimization
MDR	Mission Definition Review
MEL	Mass Element List
MEMS	Micro-Electro-Mechanical System
MOE	Measure Of Effectiveness
MOP	Measure Of Performance
MLS	Moving Least Squares
MOGA	Multi Objective Genetic Algorithms
MPMD	Multiple Program Multiple Data
OCCT	Open CASCADE Technology
OCDS	Open Concurrent Design Server
OCL	Object Constraint Language
OOSEM	Object Oriented Systems Engineering Methodology
ORR	Operational Readiness Review
OUU	Optimization Under Uncertainty
OWL	Ontology Web Language
PBS	Product Breakdown Structure
PDES	Product Design Exchange Specification
PDM	Product Data Management
PDR	Preliminary Design Review
PIDO	Process Integration and Design Optimization

PLM	Product Life-cycle Management
POD	Proper Orthogonal Decomposition
PoF	Probability of Frequency
PRA	Probabilistic Risk Analysis
PRR	Preliminary Requirements Review
PSE	Problem Solving Environment
PSS	Product Specifications and Standards
QMU	Quantification of Margins and Uncertainties
QR	Qualification Review
QUDT	Quantities, Units, Dimensions and Types
QUDV	Quantities, Units, Dimension and Values
RAM	Reliability, Availability and Maintainability
RBDO	Reliability-Based Design Optimization
RBF	Radial Basis Functions
REMS	Reconfigurable Multidisciplinary Synthesis
RIF	Requirements Interchange Format
ROM	Reduced Order Models
RoR	Ruby on Rails
RMI	Remote Method Invocation
RSDO	Rapid Spacecraft Development Office
RTI	Run Time Infrastructure
SBO	Surrogate Based Optimization
SBOUU	Surrogate-Based Optimization Under Uncertainty
S/C	Spacecraft
SE	System Engineering
SEA	Systems Engineering Advancement
SEIM	Space Engineering Information Model
SERDL	Space Engineering Reference Data Library
SME	Small Medium Enterprise
SMM	Science Margin Model
SMSWG	Systems Modeling & Simulation Working Group

SNR	Signal to Noise Ratio
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SOGA	Single Objective Genetic Algorithm
SOP	Second Order Probability
SoS	System of Systems
SPMD	Single Program Multiple Data
SSRDB	Space System Reference DataBase
SRR	System Requirements Review
STEP	Standard for The Exchange of Product Model Data
SysML	System Modeling Language
SVD	Singular Value Decomposition
TAS	Thales Alenia Space
TDD	Test Driven Development
TMM	Thermal Mathematical Model
TOR	Terms Of Reference
TPM	Technical Performance Measurement
TRL	Technology Readiness Level
TR-SBOUU	Trust-Region Surrogate Based Optimization
UAV	Unmanned Aerial Vehicle
UML	Unified Modeling Language
UQ	Uncertainty Quantification
VSD	Virtual Spacecraft Design
XDE	Extended Data Exchange
WOA	Web Oriented Architecture
XMI	XML Metadata Interchange
XML	eXtensible Markup Language
WAN	Wide Area Network
WBS	Work Breakdown Structure

Chapter 1

Introduction

1.1 Definition of the problem statement

This work was mainly animated by the need to proper manage the high number of variables and data that characterize the advanced phases of a project. Nowadays the increasing number of system complexity, considering the high number of people involved, procedures and tools, make the product lifecycle difficult to control. The effective monitoring of all the features that span from the development phase until the dismissal one play a key role within the context of the current market conditions. An efficient management of the available resources and a clear overall perspective provide the basis for the generation of product that potentially shows better behaviour. For this reason different methodologies have been recently considered to improve the systems performances, reducing both the costs and the time required to be delivered to the customer. In particular new system lifecycle methodologies have been analyzed in contrast with the traditional ones with the final aim to better manage system complexity. The correct evaluation of system complexity is in fact one of the most difficult activities that must be properly managed to avoid wrong estimation of performances and product behavior ([134]).
The Model Based System Engineering (MBSE) paradigm recently seems to be the right choice for an efficient management of all the phases that characterize a system, considering also the people and the procedures that are involved on different levels. This work considers the integration of such model philosophy with a multidisciplinary design optimization framework. In particular a space system reference case has been chosen among all the possible ones that with this methodology may be however faced.

1.2 Motivation of the choice

First of all the choice of the MBSE philosophy for the management of a complex engineering problems is strictly related to the capabilities that the related methodology allows to exploit. The overall infrastructure has been conceived following the main guidelines for the definition of a model based architecture. The object oriented approach enhances both the modeling and the analysis activities basically performed during the development of complex systems. In this context the integration with Multidisciplinary Design Optimization (MDO) techniques has been investigated to understand the current issues that prevent the application within a model based environment. The potential benefits that can be achieved through such integration are the main reason for the assessment proposed in this work. Such topic is currently not well investigated and different research initiatives are working on different directions. The correct formalization of the approach used as well as a report activity of the main involved concepts can help to paint a cleaner overview. Such information can ideally be used for future developments, paving the way for an innovative methodology for the management of complex engineering problems. The choice also of a space system was animated by the need to well represent a scenario that shows an high level of complexity, involving a great number of people, procedures and disciplines. In this way the main scope was to understand the actual benefits and to show the deficiencies that may be improved for such an approach. The reference case chosen as other similar complex systems allows to test all the functionalities and data flows that are

considered within this work.
The effective investigation of system performances is in fact one of the most challenging research activities that characterize the development of complex products and the design process of aircraft systems represents an example of such situations, where innovative solutions and approaches are continuously assessed to further improve the current methodologies ([130] and [131]).

1.3 Purpose of the proposed analysis

The purposes of the proposed analysis is to demonstrate how design variables can be monitored in a clearer way with respect to the traditional design approaches, reducing the possibility to neglect some design configurations that enhance better behaviour. This study is addressed mainly towards to evaluation of the system model methodology and data exchange between different domain-specific environments. This last feature is particularly related to the definition of a multidisciplinary design context where the close interaction between different modeling philosophies strongly affects the overall system performances. In this way a well-defined system model architecture allow to improve the whole design processes with interesting results on final product. MDO methodologies integration within the proposed system model framework is evaluated to show the feasibility and the benefits of this MBSE approach. In the last few years MDO methods have been widely used for the evaluation of conceptual configurations and system architectures. A large number of research projects is currently addressed towards the evaluation of aerospace systems performances, considering the interactions between different engineering domains and exploiting different approaches for the optimization of aircraft products (example like [80] can be found in literature for a wide range of engineering problems).

1.4 Background

Conceptually speaking the definition process that affect the product development and realization is briefly expressed through characteristic phases that historically have covered important roles. Neglecting the complexity related to the process details and all the different approaches that can be taken, the design phase can summarized with few steps. After an initial phase of requirements definition we can find the step associated to the concept creation and selection. This phase is characterized by a more deep involvement of creative skills than any other following activities. Once the conceptual design has been defined (obtaining for example multiple conceptual baselines) is possible to perform the preliminary design creation with the final aim to select one particular baseline. The following target is to reach a detailed design before the implementation of the production baseline. All the presented phases require the definition of analysis activities that allow verifying if the system design meets the requirements initially established. Proceeding through the design process the methodologies applied to deal with the problem statement is the same from a high-level perspective. The main difference is related to the level of detail that is addressed in the analysis models and the degree of interaction between the disciplines involved. One of the most important activities is represented by the problem decomposition. The models interactions and the close coupling between simulations that traditionally belong to different domain-specific disciplines demands more efforts as the project proceeds. The phases are less set in sequential way but increasingly carried out concurrently to reduce the development time. An example can be represented by a spiral product design process. In this case the concept design, system-level definition, detailed representation, integration, test and planning are all activities accomplished with an higher level of concurrence than in the past (where the traditional approach is pointed out as phased, staged or waterfall product process).

1.4.1 Engineering Design Process

Conceptually speaking the engineering design process is the set of steps that a designer takes to go from first, identifying a problem or need to, at the end, creating and developing a solution that solves the problem or meets the need. These steps can be summarized in the following list:

1. Problem definition
2. Background research activity
3. Requirements specification
4. Alternative solutions creation
5. Best solution identification
6. Development work activity
7. Prototype building
8. Test and Redesign

The current design practice is characterized often not by a sequential proceeding through these steps. Solving an engineering problem requires generally back and forth transitions between the various phases. The complexity of design process is difficult to manage in a sequential manner as system details increase and it is a common situation to return back to earlier states. Such an iterative way to work is currently well rooted in the current engineering development process. Whatever is the result of such work the creativity plays a key role within such context. The first three steps are mainly related to the modeling framework where the representative model of the system under development is defined. The alternative solutions creation is another important phase of the overall process and their role is also strictly linked with the modeling activities. The interesting element of such step is represented in particular by the relationships between the various alternative solutions with the nominal one and by their management. It is assumed also that generally a nominal configuration represents the current chosen solutions over the available ones. These are already considered and not ruled out a priori since the system is however under development and some solutions may be under evaluation, for example because the related analysis are running. A well-defined process for the management of all these features is currently one of the most challenging research activities. The potential capabilities of a well-organized and consistent procedure can help to better monitor and support the product development, providing the base for an effective way to manage the information. The present work is addressed to the conceptual definition of such phase since the alternative definition at this stage lays the foundation for the activity that characterizes the following step. The other interesting process that often plays a key role is the identification of the best design solution. The improvement of the overall system performances depends strictly on this phase. This phase is the other element that is considered in the current work. In particular a model based methodology has been proposed for the management of all the activities that orbit around such fundamental step. All the remaining items of the list are equivalently fundamental for the development of an efficient product but they are not directly covered in this work.

1.4.2 Engineering Analysis Process

The main features that are generally considered for the overall product evaluation are represented briefly by performances, costs, schedules and risks. Performances measure how well the system is able to accomplish to the primary target (mission statement for example in the aerospace industry). Costs include the development and operation life cycle resources. Schedules are instead related to the time required to implement the first unit, production rate and also all the possible activities needed to make the system ready to work. Finally the risks deal with the technical and financial failures that may be encountered. One of the first main important phases related to the product definition process is covered by the Computer Aided Design (CAD). After the initial work of conceptual identification of the possible solutions, CAD helps the designers to create a well-defined system representation. In this way we are able to clarify any doubts avoiding misunderstandings just during these preliminary activities. CAD tools assist the product development also during the following detailed process when it is used as one of the principal instruments for the configuration management, exchange of information and reference for the simulation analysis. This

element plays an important role and for this reason a multidisciplinary approach often integrates it. Another important step that characterizes the system definition is represented by the group of activities that involve Computer Aided Engineering (CAE) analysis processes. Within this category we include methodologies like FEM for solid mechanics or CFD for fluid dynamics. Other simulations that allow evaluating system behavior are also contained within this subdivision (as for instance electromagnetic simulations). Generally speaking these methods help to understand if there are design errors before the physical realization, evaluating also different alternatives through numerical simulations (some of which are listed in the following lines):

- Finite Element Method
- Boundary Element Method
- Finite Difference Method
- Finite Volume Method
- Mesh-less Method

Many engineering problems are represented with governing equations and boundary conditions. From these ones can be set and solved problems linked to mechanical or thermal field, allowing addressing also electromagnetic and fluid flow phenomena. The results generated during these phases are the key-points for the following design possible reconfiguration, representing the starting point for product optimization. Particular attention must be placed in the problems set-up in order to ensure the correctness of the data generated by the computer.

1.5 Problem Solving Environments (PSE)

The management of complex problems becomes particularly difficult when a wide range of engineering domains are involved within the design process. The solving tools, methods and process are often not so easy to handle for different reasons. The codes or models could be written few years ago by people that are no longer working on the same subject for example. The use and maintenance of the available resources (as those coming from company knowledge) becomes difficult and in such cases a Problem Solving Environment (PSE) represents a well suited solution. A PSE is basically specialized computer software that is mainly used to solve one or more class of problems. Such objective is obtained through the combination of automated problem-solving techniques with user-oriented tools conceived to guide the problem resolution. The first examples of PSE were born in the 1990s and initially they were built with the same language of the related field, employing often a graphical user interface with the solving code. The purpose is the definition of an interface that enables other users to manage a domain specific-software. The first prototype was generically used above all for data visualization or representation of large systems of equations. In the next years they were improved to be also used in the management of narrow field of science or engineering. In this way a gas-turbine design code could be implemented to simplify the access and use of the available resource for example.
A PSE generally provides all the computational facilities that are needed to approach a target class of problems. In particular these features include advanced solution methods, automatic and semiautomatic selection of solution methods and also ways to easily integrate new solving techniques. Moreover, PSEs use the language of the target class of problems, so users can run them without specialized knowledge of the underlying computer hardware and software. By exploiting modern technologies such as interactive color graphics, powerful processors and networks of specialized services, PSEs can track extended problem solving tasks and allow users to review them easily. Overall, they create a framework that is all things to all people: they solve simple or complex problems, support rapid prototyping or detailed analysis and can be used in introductory education or at the frontiers of science ([28]).
PSEs can basically be considered all the computing systems and infrastructures that are conceived to help

computational scientists get their work done in a more effective way. Such environments include generally all the features needed to support the problem-solving activity, from problem formulation, algorithm selection, numerical simulation and solution visualization. They are also defined to provide useful capabilities to improve the collaboration among people separated in space and time, often using different set of codes and machines. Computer Aided Engineering (CAE) is one of the most important Engineering field and some quite sophisticated PSEs have been developed to support the related activities. All the current PSEs follow basically the conceptual guidelines previously introduced and the related implementations are based on the specific needs of the developed framework. The same solving approach can in fact be actually implemented through different architectures, as a desktop or web-based solution for example. The choice between the various alternative solutions depends strictly on the primary goal of the research activity where the subject has been defined. In literature different PSEs research activities a prototype can be identified, each one addressed to a particular class of problems or conceived and customized on the basis of the required capabilities. In the following lines some example will be briefly introduced.
W-DPSE represents one of the prototypes that have been developed to assess the capabilities of a PSE as useful framework to support CAE technologies. The W-DPSE name stands for web-based distributed problem-solving environment and has been conceived to provide an effective approach to distributed modeling and simulation, paving also the way for networked collaboration. The main objective is to provide a tool that can be interactively used to explore and visualize the design work activities. This system is built as a three-tiered architecture represented by three main layers: a web client presentation interface (WCPI), computing solver servers (CSS) and a system management server (SMS). All the related components of this infrastructure are implemented with an object-oriented approach using Java as programming language while the remote method invocation (RMI) technology is used to communicate across the layers. In particular the developed framework includes efficient interface for wrapping legacy computation codes or interdisciplinary and diversified applications defined for example in C, FORTRAN or other languages. Such objects are wrapped and provided as Java component through the implemented framework. The communication mechanisms between Java component and legacy codes are defined through java native interface (JNI) and UNIX inter-process communication (IPC) by the way of operating system. A more detailed description of such PSE framework is available on [29]. In this case the PSE framework has been mainly conceived to provide useful interface for already valuated and tested solving codes. A well-defined interface, not only solvers are, respectively independent from the both servers and clients but also clients and servers are isolated. In this way clients can use the capabilities of the servers without a specific knowledge of the server architecture and communication protocol. In particular users can create their own models (for the available solver implemented) through the use of a registered model generator. The user can also perform remodeling once for example the analysis responses have highlighted any strange or unexpected behavior. In this case a new input model can be created and submitted to the system for new analysis, paving the way to re-design activities and iterative development processes. Interesting results are also provided in [30] where a research activity has been addressed to the evaluation of a PSE portal for Multidisciplinary Design Optimization. This PSE infrastructure has been conceived to face one of the main problems that characterize the application of MDO techniques in the context of a complex project. Applying MDO methodologies in real engineering problems requires the user to spend a lot of time arranging and interfacing resources used in the process. In this case a web portal provides useful utilities for the management of models and resources within a shared environment. The actual implementation is based on Globus toolkit version 4 (GT4) web service-based technologies for distributed middleware, mainly used for the transmission of a large amount of data. This toolkit is basically constituted by a series of libraries and programs that handle the general problems regarding the definition and implementation of grids and distributed systems. In particular three containers can be identified in GT4 and they are represented by a Java container, a C container and a Python container, using the services developed respectively by these ones. The standard protocol technique used is based on eXtensible Markup Language (XML), ensuring the independence of the portal from the platform and the programming language. In this case the user can define the overall process through a process definition service provided by web interface, editing, storing ad correcting the related process resources. The created process is managed in background as XML format, becoming also read to be executed. The user interface provides basically five different capabilities summarized as: problem

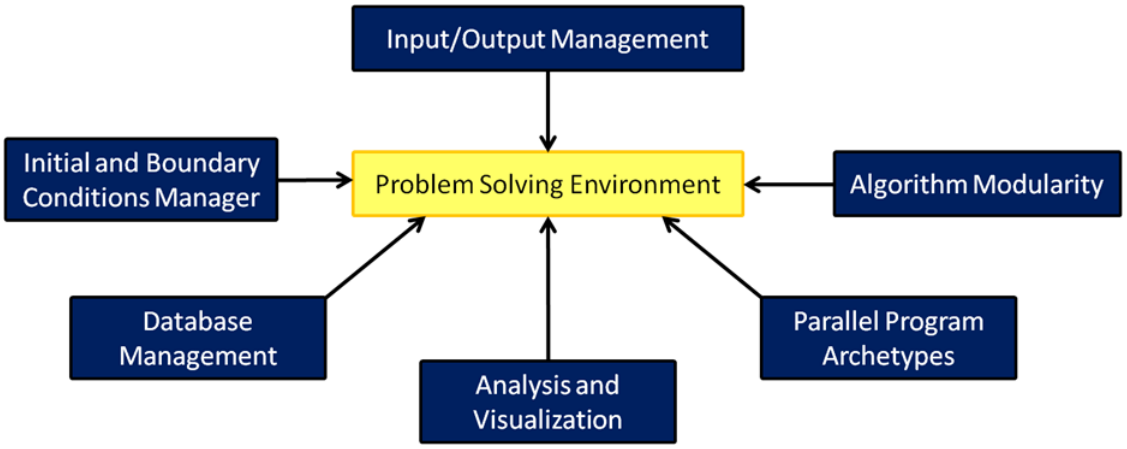

Figure 1.1: Example of the aspects that can potentially affect the definition of a Problem Solving Environment.

description management, security management, data management, resource management and workflow management. Globus toolkit functionalities allow providing such web services through a Simple Object Access Protocol (SOAP) web service technique while the connection with the database is realized through Java Database Connectivity (JDBC) technology. The design resources provided by the implemented framework are represented by analysis codes, optimization codes and CAD objects. The process flow for the development of MDO framework as provided by the portal can be summarized in the following sequential steps. First the design object must be selected and then the related design resources must be identified. These ones can be chose among the object stored in a resource repository where the elements can be saved after a proper registration. Once the resources have been selected the next step is represented by the workflow process definition followed by the input-output variables linkage. This activity is done through the utilities provided for the management of data connections, allowing the correct associations between the available variables. Once these phases have been accomplished the database is created and an MDO framework is created, potentially ready for design phase. The implemented environment provides all the elements required for the definition of a Multidisciplinary Feasible (MDF) method on a specific problem. The MDF is an optimization method that integrates the design resources as a single design process (conceptually al the solving codes and models are linked sequentially).
An example of the aspect that cab be related to the definition of a problem solving environment are reported in figure 1.1.

PSEs frameworks have gained increasing importance in the field of aerospace design process, above all in the last few years. The developments of new software methodologies, advanced approximation methods, data storage and fusion techniques as also the improvements in computational hardware have driven a deeper integration of such technologies within the development process with respect to the traditional design approach. Such improvements have lead in fact to conceive new ways of manage the design process of complex systems, all the related features and organizations. A well-documented evaluation about the traditional design approach in aerospace field and the increasing needs are available in [31]. The same article provides a clear list of the improvements that can be identified in the design process and the related technologies:

- Improvement of the quality control
- Support of the team decision making process
- Improvement of the design environments
- Creation of a seamless integration between design and analysis
- Understanding of the product realization process
- Storage and re-use of design history

- Determination of the impact of decisions
- Promotion of continuous learning
- Integration between analysis tools
- Enhancement of creativity and innovation
- Reduction of the development time by increasing parallelism
- Improvement of the information infrastructure
- Production of globally optimized designs
- Management of complexity and risk
- Enhancement of the critical thinking and evaluation methods
- Integration of product design data
- Improvement of communication of design specifications to remote sites and companies
- Integration of product manufacturing process development
- Integration of large-scale systems

The enhancement of creativity and innovation is one of the main interesting and challenging feature related to the proposed ones since it is directly related to the development of new solution and technologies for the market.
The traditional software environment is based on the functionalities provided by a corporate intranet at which the user workstation is connected. Corporate CAD systems (as commercial solutions and Product Life-cycle Management (PLM) infrastructures, including in this one also the Product Data Management (PDM) system), analysis software (for structural, costing or performance computations) and computing node are all connected to the same corporate network. This environment can be improved integrating more infrastructures with the final aim to paving the way for the definition of a PSE framework. Different user's workstations cab be connected with a team leader workstation and all ones can be linked a multimedia and virtual reality system. A corporate network as a can be used to connect such workstations with a data-base master, a design process monitor, computing grid and an optimization system. Database master can be configured to manage both the design data-bases (containing the current project information) and a design archive (storing all the available and accessible data coming from previous project) for example. This example represents one possible conceptual solution for PSE architecture but other different configurations can equivalently be chosen. The PSE architecture considered in [31] is basically represented by a Wide Area Network (WAN), which represents the corporate network on which graphical user interface is used to manage for example the computation nodes. CORBA wrappers are instead used to integrate the computational software and resources on the same network, providing all the required utilities for the management of system design.
These examples show how the application of web-based technologies can help and support the analysis activities. In particular the developed frameworks have been mainly addressed to the execution of simulation scenarios providing a graphical user interface for the management of the available resources. The user interface has been conceived to handle already defined models and simulation codes in the large part of the considered cases. The management of analysis resources not already registered in the same system make the overall framework difficult to realize. This situation represents a challenging problem and different solutions can be considered for the right evaluation of the possible approach. In particular one of the current integration issues that limits the capabilities of PSEs frameworks is represented by a correct integration between a system modeling environment and analysis ones. The objective of this work is mainly addressed to the assessment of the possible connection between a modeling environment and analysis

infrastructure. In particular their integration will be based on web-based technologies since such choice has highlighted interesting results in the case for example of already developed PSEs, as briefly introduced in the previous lines.

Chapter 2

System Engineering

System engineering is currently gaining an increasing key-role within the design process of complex products. Generally speaking it represents a multidisciplinary approach addressed to the development of balanced system solutions with respect to different stakeholders needs. This balance involves both the management and technical processes with the main aim to reduce the possible risks that can affect the success of a certain project. Management activities are mainly addressed to the monitoring of development costs, schedules and technical performances, ensuring that the project objectives are met. All this processes are deeply related to the managing risk and decision making activities. On the other side the technical process are mainly related to the specification, design and verification of the system to be build. Technical processes can be summarized with the following conceptual activities: the system specification and design, the system integration and test, and finally the component design, implementation and test. All these simplified class are strictly interrelated and iteratively applied during the development of the system. Some of the most important activities that cover a fundamental role are reported in the following list:

- Elicit and analyze stakeholder needs
- Specify system
- Synthesize alternative system solutions
- Perform trade-off analysis
- Maintain traceability

Two of the most interesting and challenging phases are represented by the capability to synthesize alternative solutions and perform trade-off analysis which are also mainly discussed within the present work. A clear understanding of the stakeholders needs is one of the complex phases since the decisions made during this early definition process can heavily affect the effectiveness of the final product. It is extremely important to understand how the external systems, users and physical environments are interfaced with the system itself to clearly demarcate the boundary of the system and the associated interfaces. This process may also be characterized by a well definition of the functions that have to be considered to be compliant with the consumer requirements (functional analysis), specifying their sequence and ordering. Once certain specifications are made the following phase regards the design of components and their test, providing the right feedbacks to the system specification process. In this way the design evolves iteratively towards the definition of the final system solution. It is important during this process to well define the information flow that starts from the stakeholder needs down to the components requirements. System representation often includes a wide set of stakeholder perspectives, involving the participation of many engineering and non-engineering disciplines. A typical multidisciplinary system engineering team should include viewpoints from each of these perspectives and people coming from different domain-specific fields that have to work together within a system that is increasingly complex and where all the various disciplines are deeply integrated. The complexity of the systems considered often drives towards the definition of a System of Systems (SoS) structure. This viewpoint is based on the identification of an element as

the part of another system on a higher level of definition. The need for the correct management of system complexity has lead to the definition of a various standards as support for the different perspectives that characterize a certain project. In particular different systems engineering standards are matured over the last several years with the main purpose to reduce as much as possible the errors related to the data exchange between different environments. Some of the possible system engineering standards are reported in the following list [1]. Process standards:

- EIA 632
- ISO 15288
- IEEE 1220
- CMMI

Architecture framework:

- FEAF
- DoDAF
- MODAF
- Zachman FW

Modeling methods:

- HP
- OOSE
- SADT

System modeling standards:

- IDEF0
- SysML
- UPDM

System simulation and analysis standards:

- HLA
- MathML

Interchange and metamodeling standards:

- MOF
- XMI
- STEP/AP233

It is important within this context to provide useful definitions for the concepts that are widely considered in the present work, highlighting in particular the terms of engineering processes and methods. Generally speaking the term system engineering process identifies what activities are performed during the project but not give details about the ways they are performed. The system engineering method defines instead how the various activities are performed, describing the types of product that have to be obtained and how they are designed and developed. Another important feature is represented by the concept of operations which defines how the system interacts with the external environments and how it has to behave from the stakeholders' perspectives. The main objective of the modeling standards is represented by the identification of a common language for the description of system physical architecture, behavioural models and functional flow. Model and data exchange is one of the most challenging and critical activity during the development process, above all when different domain specific tools have to interface for the data sharing. The XML Metadata Interchange (XMI) specification has been conceived within the context of OMG and has the purpose to support and make easy the model data exchange when MOF-based languages are used (such as SysML or UML). In the same way the Model Driven Architecture (MDA) paradigm is addressed to the definition of further standards, ideally enabling the transformation between the models and different modeling language. All these efforts are addressed towards an improvement of tool interoperability, modular modeling process and reuse of system design product, reducing the time and costs related to the implementation of already defined objects.

2.1 Lifecycle management

In industry, lifecycle management stands basically for product lifecycle management (PLM) and all the related concepts must take into account such definition. PLM can be defined as the process of managing the entire lifecycle of a product from the initial idea to the following phases of design, manufacturing, operative service and final disposal. Product lifecycle management basically integrates people, data, processes and business infrastructures, building up the product information backbone for companies and their enterprise. Lifecycle management processes can be characterized by slightly different phases that show different time extensions and conventions but they are all conceived to organize the work from the preliminary steps to the more detailed ones. The brief introduction about the current lifecycle management process descriptions allows better understanding the context for the following work. The present research activities is developed starting from the actual lifecycle management process strategies with the final aim to propose and evaluate a model based modeling and analysis infrastructure. This concept requires a well clear view of the system engineering methodologies for the management of product development from the early phases to the more advanced ones, until the final disposal. Figure 2.1 conceptually reproduces the activities and related relationships that generally characterize the overall process from customer needs to the final system solution.

Such concepts and their correlations can however better explain through other diagrams and representation models. In the last few years large-scale system projects have been created through the use of different lifecycle development models. There are no particular constraints on the development model that must be used and organizations, academia and industry often use their lifecycle patterns also if three main typologies can be identified. At the moment such lifecycle development models are summarized by Royce's Waterfall Model [2], Boehm's Spiral Model [3], and Forsberg and Moog's "Vee" Model [4]. All such models approach the definition of lifecycle in different manners as shown in their related conceptual representations in figure 2.2, 2.3 and 2.4. Such lifecycle model representations are partially derived from the patterns used also to implement software product and the same approach can also be applied and extended to the development of complex systems.

The definition of lifecycle development process through V-diagram allows to graphically describing the overall process of system design and manufacturing. This representation can be used to equivalently reproduce the same conceptual process at different details levels since the same structure can be adopted to define the whole system as also a single subsystem or component. The same diagram can in fact be applied at different detail levels to show the process of design and manufacturing, providing a visual organization

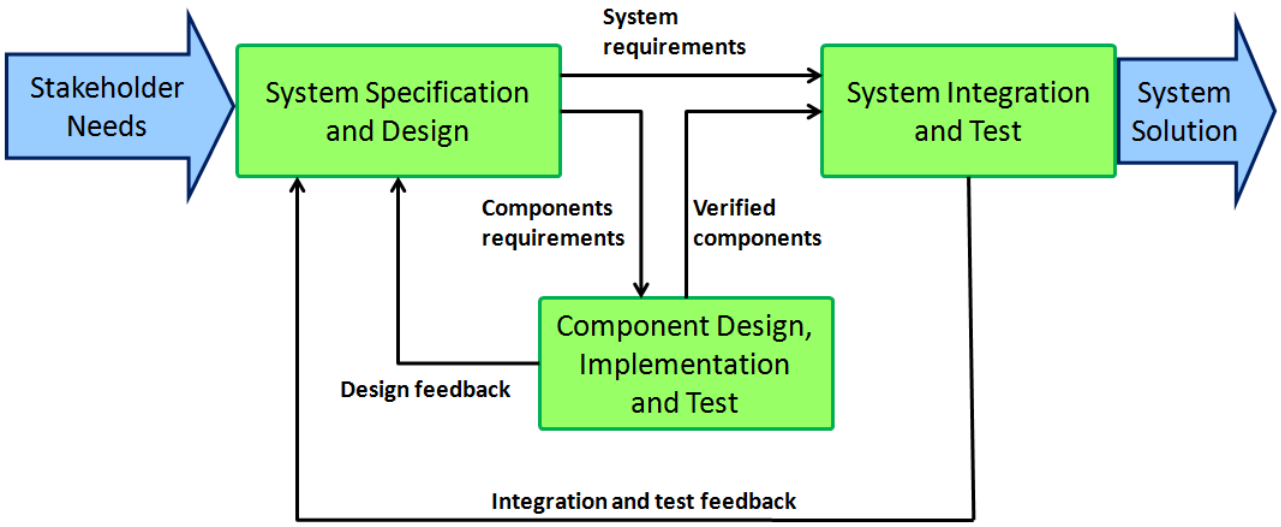

Figure 2.1: Development process from customer needs to system solution.

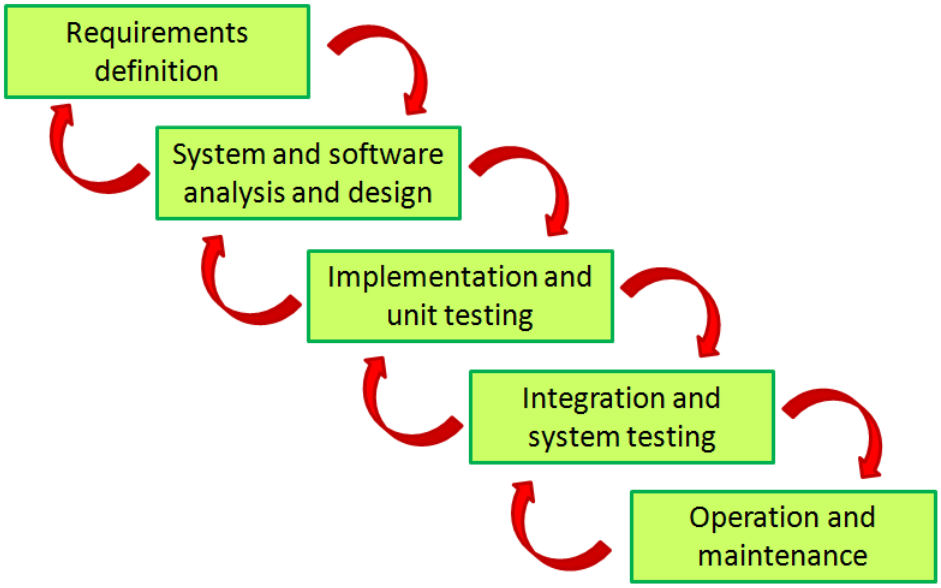

Figure 2.2: Royce's Waterfall Model.

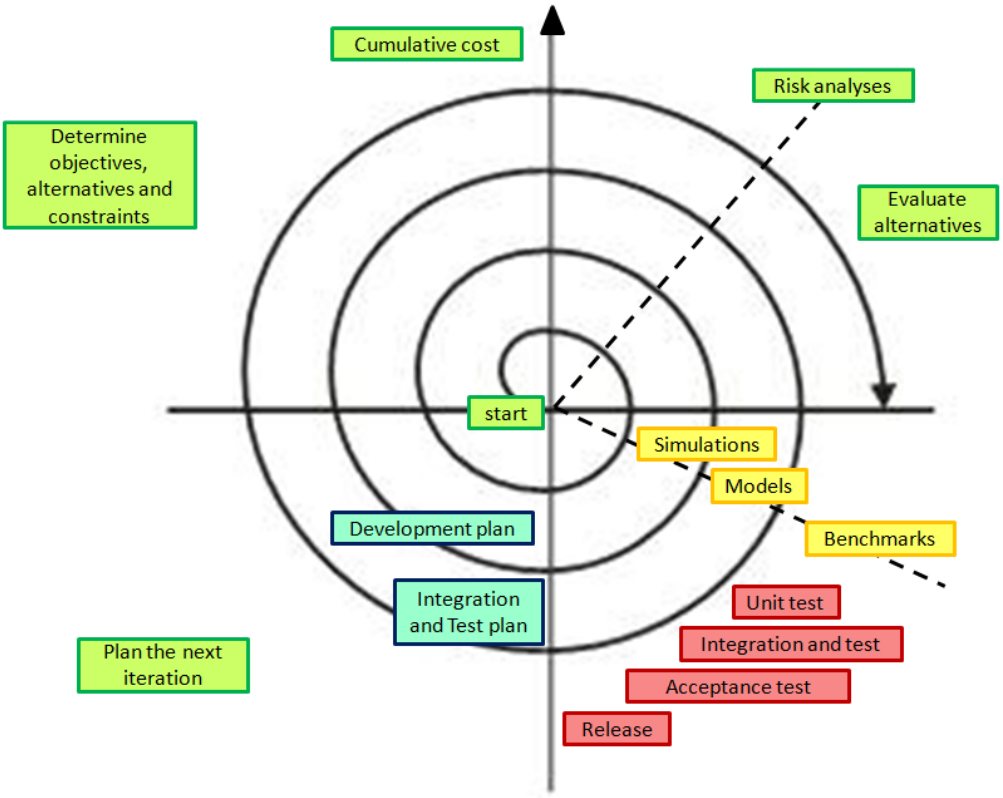

Figure 2.3: Boehm's Spiral Model.

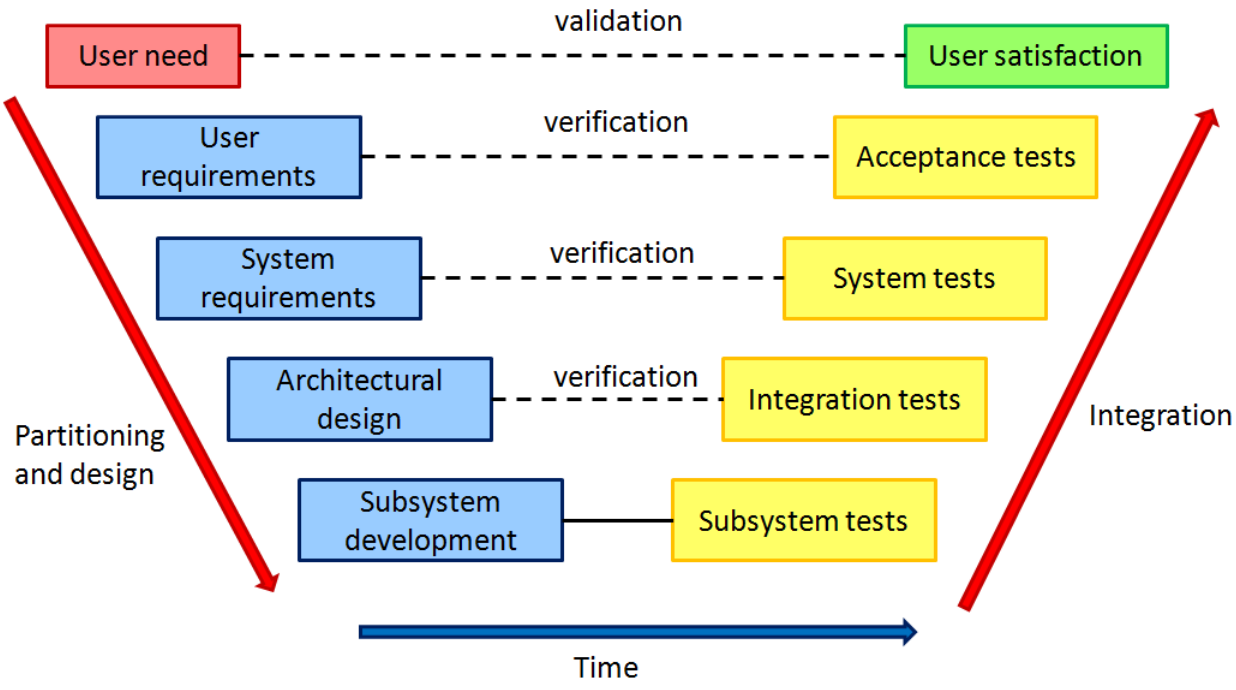

Figure 2.4: Forsberg and Moog's "Vee" Model.

of activities as the integration, testing, verification and validation for example.

2.2 System Analysis concepts, methodologies and activities

Models creation from mathematical relationships and physics-based rules is now one of the most interesting research topics. In particular the information gathered within the system data model may be used to properly define its virtual representation. The development of this relation can be realized under different approach, depending on the required information. In particular in the last few years some modeling infrastructures and languages have been developed. An example of a promising modeling languages that has started to spread across different engineering domain is represented by Modelica. In particular it is a well suited language for the characterization of the system behavior, providing useful capabilities over a wide range of applications in the field of system analysis. The equations related to a particular element of the system could be used for example to set the physical laws that are successively used to build up and manage virtual simulations. The main issue concerns about the translation of the involved equations into useful codes that may be processed in the right way.
In the following subsections some example of the most widespread analysis activities and other associated concepts are reported to better describe the analysis process that characterizes the overall system.

2.2.1 Uses cases and Scenarios

Analysis activities are often characterized by the clear understanding of product use cases as well as the correct identification of the related scenarios. Such concepts must not be confused and a much more detailed description of these terms will be provided in the following sections. These words have two different meanings in the context of System Engineering and they must be understood to avoid future misunderstandings.
Use case is defined as a group of scenarios linked together by a common user goal while a scenario can be defined as sequence of steps that describe the interaction between an actor and a system.

2.2.2 Requirements Analysis

One interesting aspect that is strictly related to the methodologies of System Engineering is represented by the requirements analysis. In the preliminary phases a clear understanding of requirements, their significance and relationships are fundamental step for the right start of the development process. Customer needs provide the information from which the project requirements are built, ensuring the definition of such guidelines that drive the design. The correct capture and analysis of system requirements cover a basic role and different methodologies can be considered for such activities. System requirements can basically be distinguished between functional requirements and non functional ones. In the first case the requirements refer directly to those functions that system must perform, such as doing particular actions and activity or showing certain capabilities. These requirements are generally not linked to numerical quantifiable properties. Non functional requirements are instead represented by those specifications that can be expressed or traced to numerical values. Performances requirements belong to this second category for example. This classification can be further detailed but such distinction provides enough details for the main purposes of the present work.

2.2.3 Functional Analysis

Functional analysis covers a key-role for an effective development of complex products and represents one of the main pillars of System Engineering discipline. Such activity is mainly addressed towards the identification of all the functions that the product must perform during its operative lifetime. The right definition of these functions and their relationships with the product elements is particularly important to allocate the resources that will be provided by the system. This analysis do not involve all the engineering

domains at the same level or during the same lifecycle phases. Some disciplines exploit functional analysis to mainly support the preliminary development phases while other ones are characterized by this activity much more extensively during their processes. For example Mission Operations discipline is basically affected by the results coming from functional analysis which plays a fundamental role for the correct identification of the interactions between actors and product components (procedures definition). The main aim of functional analysis can be summarized by the collection of all the activities that are animated towards the clearly characterization of what the product is able to do. It is important not to confuse such concept with that related to the Operational Analysis which can be slightly similar with respect to some activities but are basically conceived with two different purposes.

2.2.4 Operational Analysis

The Operational Analysis is another important activity of System Engineering domain that generally characterizes the development process of complex products. The main purpose of such analysis can be summarized with the identification of how the system behaves mainly during its operational lifetime. In this case the main emphasis is not on the functions that the system is able to perform (aspects handled with functional analysis) but mainly on its states during one of the possible operational scenarios. This analysis includes for example the activities directly regarding state machine modeling for the system under development. In this case it is more important to understand the relationships between the possible states in which the system can be as well as the events that regulate the transitions among these ones. This activity can help to get a clear vision of system behaviour, providing the instruments to support the investigation of the possible combinations of a complex situation. Operational analysis covers a fundamental role for Mission Operation domain as well as the functional analysis. The correct identification of system states supports the proper scheduling for the activities that can be performed by the product. The procedures that the users must follow to rightly operate the system are directly made from the output coming from the operational analysis. Power budget represents an example of the possible evaluations that can be basically performed starting from the data provided by operational analysis.

2.2.5 Cost Analysis and Estimation

An important definition that mainly covers a key role in the evaluation of system costs is represented by the Work Breakdown Structure (WBS). Such term refers to the hierarchical decomposition of the work necessary to complete a project/program. Such breakdown structure can also contain the Product Breakdown Structure (PBS), which can be identified with the term System Breakdown Structure with US DoD notation. Cost estimation methods can be summarized in the following ones:

- **Parametric cost models**: the estimation of project costs is achieved on the basis of equation based approach. In particular some system key parameters are used as independent variables to compute costs. Such driving variables can be represented by weight or performances indexes, ensuring the repeatability of the achieved results but at the same time the accuracy of the obtained responses is not well pursued. This method is basically used during the trade studies or however the preliminary design phases.

- **Analogy**: this method is applied when the system under development shows some similar characteristics with respect to another one that has already been developed and built. In this case the current estimation is obtained through the evaluation of the costs already known about the similar product and some correction can be introduced to take into account for little differences.

- **Grassroots**: cost estimation is evaluated through a bottom-up approach where a particularly detailed data about the project is required. Such an approach is used mainly during the advanced phases of the program.

2.3 Simulation Model - Mathematical Model

The main aim of simulation modeling is basically represented by the analysis of the nature and behavior of a particular system. Generally speaking system can be identified as a facility or process that is under consideration due to different reasons. In order to analyze the behavior of a particular system one of the most important process is represented by the making of a set of assumptions about its response to external input. The whole set of assumptions that are made to define the nature of the system can be expressed in the form of mathematical or logical relationships and all contribute to define the model characteristics and how it behaves. In this way the final objective is to build instruments that can be used to imitate or simulate the responses that we want to study. When the previously introduced relationships can be defined through the use of mathematical methods to obtain exact function about the information of interest, we are in the case of analytic solution. With the mathematical methods are referred algebra, calculus and probability theory. Real-world system often cannot be defined analytically and in these cases the simulation represents the unique feasible solution. The system behavior is evaluated through the use of numerical model in order to estimate the desired response. The same system may be modeled with different approach depending on the features that want to be studied and also from which viewpoints. Considering these characteristics the simulation methodology became one of the most import aspects of the model building phase. In the following sections a brief introduction on the main features of system modeling is described and a clear definition of the terms system, model and simulation can help to better understand the studied methodology [5]. System identifies the collection of entities, such as people and machines that interact together for the accomplishment of some final objective [this definition was proposed by Schmidt and Taylor (1970). What the term system refers to depends often on the particular objectives that were faced within a certain study. For example what is defined as a subset model for a particular system can represents the whole system under different simulation conditions. The term model is usually used for a structure which has been built purposely to highlight some particular features and characteristics of some other components [6].
Another important definition is represented by the state term. The state of a system refers to the collection of variables that must be defined to completely describe the model at a particular time. Systems can be categorized as discrete or continuous. A discrete system is characterized by state variables that change their values instantaneously at different points during temporal evolution. In the case of continuous system instead the state variables change in continuous manner during model simulation. In the real-world representation it is difficult to find systems that are wholly discrete or wholly continuous but is however possible to classify their belonging on the fact that one of the two types of state variables predominates over the other. During system analysis the need to study the relationships between some components as the possibility to face different boundary conditions drives to different ways under which system can be represented. On the basis of the features to be analyzed (for example considering the need to evaluate the performances under changed conditions) there are different ways to study a system as reported in figure 2.5.

System can basically studied starting with the distinction between experiment realized with the actual system and experiments with a model of the system. In this last category are included the physical model and the mathematical model. Another distinction can be based on the resolution approach that can be applied on the mathematical model, distinguishing between the analytical solution and simulation. The term simulation refers mainly to the numerical solution of a mathematical model. The first main distinction between experiment with actual systems and experiment with a model of the system depends strongly on the available resources. The better solution is always to experiment over the actual system to obtain more reliable information on responses to the input parameters but often this condition is not possible to realize. This situation is desirable but often the experiments over the whole system become a costly operation or in other cases the tests to be done are disruptive for the system (as for example in the case of thermal or structural tests). In other situations, at the time the experiments are needed, the system is not present or however is not possible to realize such experiments for security problems or not-repeatability of operations (such in the case of system involving nuclear applications). These reasons animated the building of a model as a representation of the system to be studied as a surrogate for the actual system. One

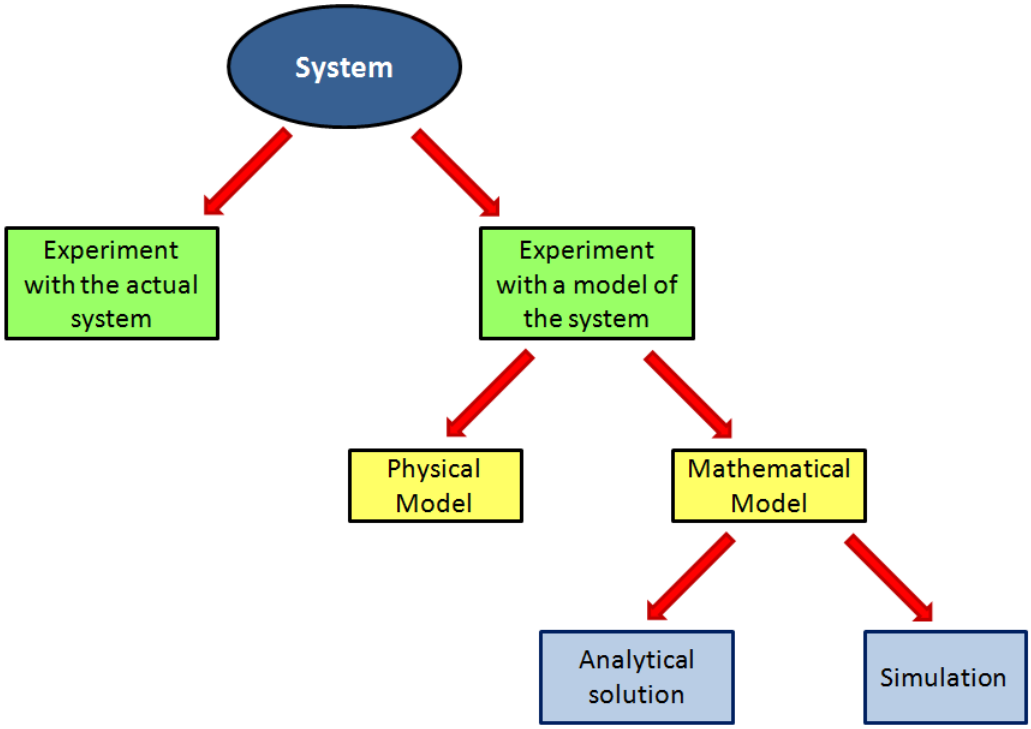

Figure 2.5: Conceptual overview of the possible ways to study a system [5].

of the problems that the modeling activities involve is represented by the need to well understand the validity of the model (referring to the capability to well model the responses that the model is defined for). The other two categories refer to the distinction between the physical and mathematical model. Physical models are represented by actual models that reflect some particular system characteristics (for example they refer to the model used in the case of wind-tunnel simulations), depending on the output that have to be monitored in that particular case. In other context the term physical model is equivalently expressed as iconic model. These models are not cost-effective for the main analysis purposes so they are often overwhelmed by the definition of mathematical models that are currently the main approach for predict system responses. These models define the logical and quantitative relationships that are manipulated to evaluate system reactions and that can be used to understand which would be the actual responses. The same classification is defined as concrete versus abstract models according to other references. The essential feature of mathematical model can be identified in the involvement of a set of mathematical relationships, such as equations, inequalities, logical dependencies, etc. The other main subdivision that characterizes the mathematical model class is represented by the distinction between the analytical solution and simulation. In the case the model is simple enough to be managed through the implementation of exact relationships between the quantities involved in the problem the solution can be defined analytically. When the relationships and equations that are directly bounded to the problem are complex, the solution of the problem can generally be obtained through numerical approaches. In this case the analysis of the response functions on the basis of the chosen input parameters are studied by means of simulation. Once what is defined as simulation model is implemented it is possible to introduce other classifications that allow characterizing other different ways of representation. In particular these classifications are represented by the following couples, generating all the possible combinations between each other.

- Static and Dynamic Simulation Models
- Deterministic and Stochastic Simulation Models
- Continuous and Discrete Simulation Models

Static simulation models are represented by those cases where the problem is defined at a certain time or in the cases where the time does not cover an important role. On the other side dynamic simulation

model refer to the problems where time parameter covers a not negligible role. Static simulations can be represented for example by some Monte Carlo models. Deterministic simulations are those characterized by the absence of any probabilistic quantities. In particular the output functions are determined once the input values and their relationships are uniquely specified, not depending on how much time the simulation lasts. When the simulation models include at least one random variable the simulation is defined as stochastic and output quantities must be analyzed through probability theory. Roughly speaking deterministic models are a special case of stochastic models as demonstration of the close correlation between the elements of this class of simulations. Discrete models are represented by the case where system under study is analyzed as discrete simulation while continuous models can be represented by the definition of variables belonging to a continuous domain. The choice between discrete or continuous modeling for the same phenomena depends strictly on the needs and the objectives that are desired or required. Another important concept that is recurring in the field of models simulation it is represented by event. This term stands in particular for the instantaneous occurrence that may change the state of the system. Mathematical programming as defined does not refer to computer programming concept while it expresses the planning activities behind the problem formulation. Most of engineering applications that involve mathematical programming are generally addressed to the resolution of optimization problems also if that is not the only activity that characterizes its implementation. Mathematical programming can be applied on different models categories as expressed in the following list.

- Linear Programming Models
- Non-linear Programming Models
- Integer Programming Models
- Stochastic Programming Models

2.4 Space System Engineering

The integration of MBSE methodology within the project of complex system has found a productive environment in the context of Space Engineering. This field has always been characterized by a high level of complexity with respect to other engineering applications. The wide number of products, people, domains and processes involved favours the creation of an environment difficult to manage and control. The developments of methodologies that can help to better organize such context are seen as extremely interesting and strategic for the right design of a correct system. The product life-cycle management (PLM) process covers a key-role for the definition of a complex space system. A comprehensive work about the formalization and definition of the PLM guidelines mainly addressed to the definition of space systems is available in [7]. Some of the main important concepts on which is based the present work are derived from the definitions and information provided by this handbook. In particular the high-level perspective on the integration between the modeling and analysis environments has been selected as primary element for the implementation of the proposed framework. In this way the developed infrastructure is consistent with the concepts already available from the system engineering practice and knowledge coming from space field. The system design main process can be conceptually represented as in figure 2.6. The schema provides a clear pattern of all the relationships that characterize the design activity of a space system. The high-level design phases can be summarized in the block representation where the product breakdown structure is identified after the requirements analysis and the functional decomposition, clearly separating activities that involve procedures, people, tools and methods different from each other. The main processes of design and product breakdown structure, functional and logical decomposition as also requirements analysis and allocation are all include within the modeling infrastructure of the proposed methodology and framework. The processes of functional and performances analysis are instead approached within the analysis infrastructure of the same environment. In this way the main idea is to clearly keeping separated the modeling activity (including processes, people, tools and methods) from the analysis one, which refers mainly to the evaluation of the design baseline already modeled.

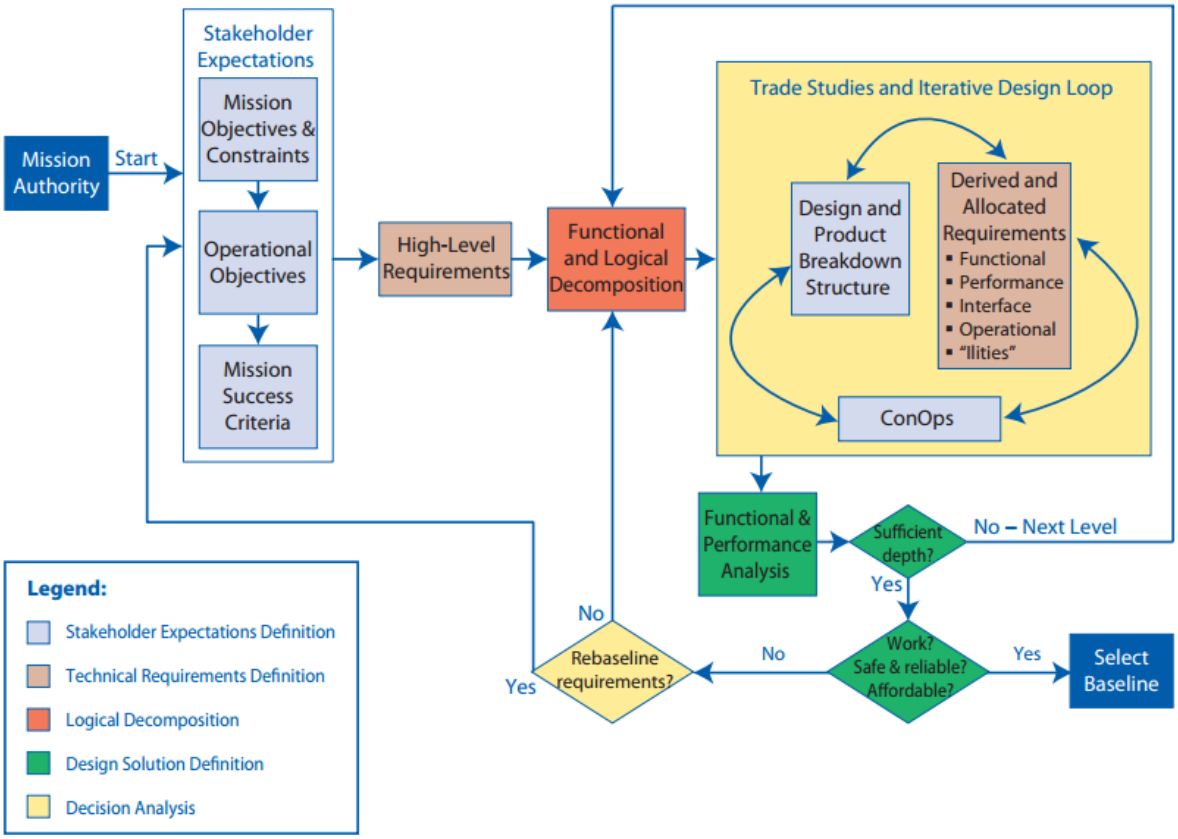

Figure 2.6: High-level representation of the main conceptual processes involved in a space system definition [7].

The product life-cycle management can be approached considering different phases interspersed with different key decision points. Their definition is strictly dependent on the knowledge matured throughout the years and for this reason different aerospace agencies have often their own lifecycle management procedures, milestones and acronyms. Some examples of such management process are briefly reported in the following, providing a high level perspective of how space systems development activities are organized [8]. The main phases are often identified with the most important key decision points and the related milestones are followed by the delivery of document and reports on the project status. All these documents contain the description of the current development level of the system, the results coming from analyses of its performances and currently opened issues. The considered timelines are used to properly allocate the available resources on the basis of the related phase, providing both the time slots for the organization of workload and useful indications for activities coordination. In figures 2.7, 2.8 and 2.9s are reported the time allocations for the various lifecycle phases.

2.4.1 European Cooperation for Space Standardization - ECSS

Some of the topics introduced in the current work have also been developed and extended from the concepts and definitions provided by the European Cooperation for Space Standardization (ECSS) organization. Such institution is mainly devoted to the coordination of standardization activities with particular emphasis on space systems. It is supported by several agencies and companies that are interested in the definition of a common set of elements, definitions and guidelines that can be shared among them. More details are available on the related web portal [9].
Until few years ago there is no uniform system of space standards and requirements in Europe. Although the presently used standards and requirements are quite similar, the remaining differences result in higher costs, lower effectiveness and in a less competitive industry.
At the beginning of 1993 the European space community realized that a solution had to be found to overcome these problems, and expressed their will to develop a new coherent system of European space stan-

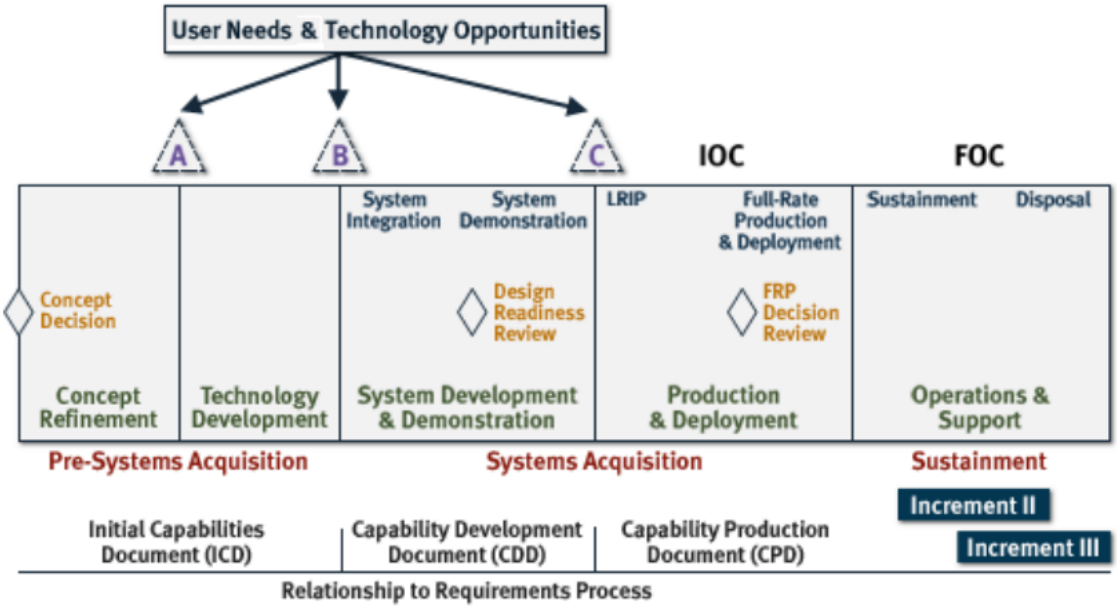

Figure 2.7: Department of Defense (DoD) Product Life-cycle Management process [8].

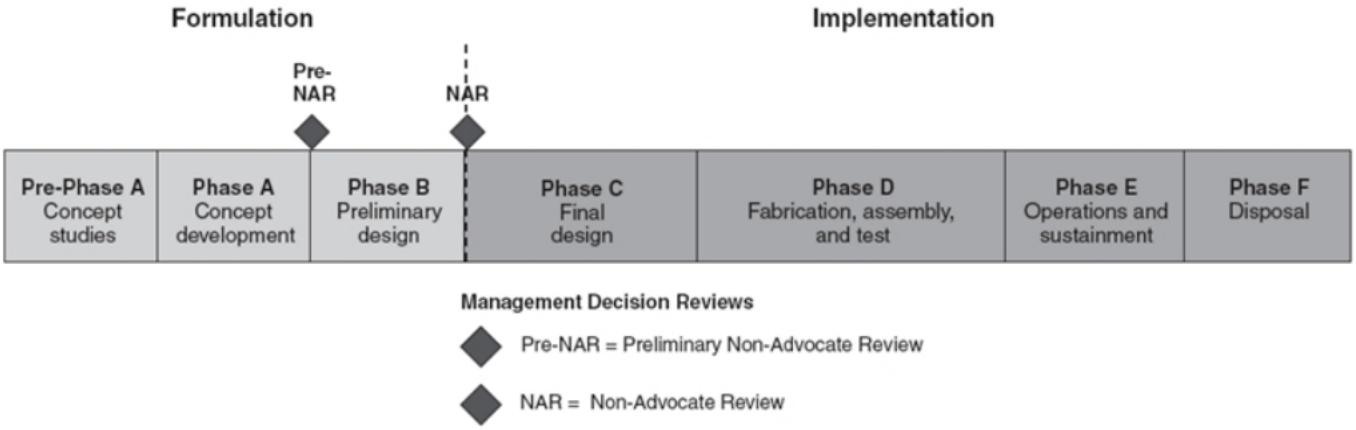

Figure 2.8: NASA Product Life-cycle Management process [8].

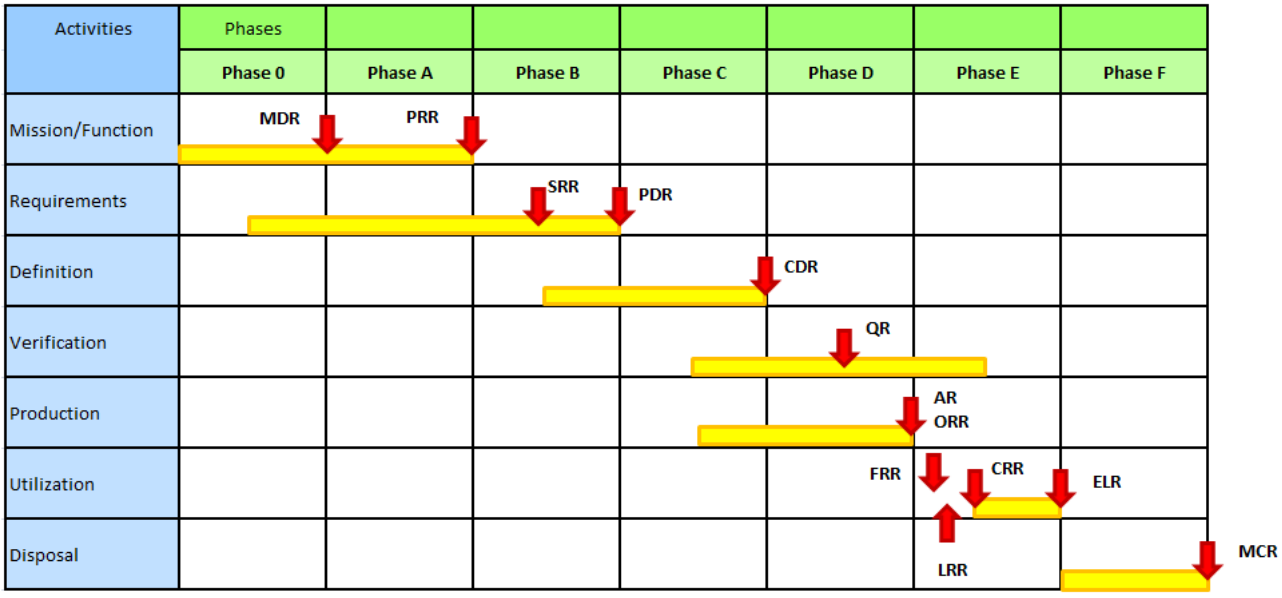

Figure 2.9: ECSS Product Life-cycle Management process.

dards.
The European Cooperation for Space Standardization (ECSS) was started officially in the autumn 1993, when the partners signed the ECSS terms of reference (TOR), which defined the framework and basic rules of the system. At this point, the partners jointly undertook the development of the system, designed to meet the main objective of providing a single coherent set of standards for use in all European space activities and particularly projects. The European space industry was fully associated with ECSS from the outset. The first task of the ECSS was to draw up a policy document. A dedicated working group was set up in late 1993, leading to the publication of a document entitled "Standardization Policy" under the number ECSS-P-00. This document reports the different aspects of the system, including scopes, objectives, implementation, authority, organization and documentation.
ECSS policy dictates, that ECSS standards shall promote the continuous improvement of methods and techniques, and the avoidance of unnecessary work. Experience from past projects and other appropriate sources shall be systematically incorporated into the ECSS system. ECSS standards must satisfy all European and international clients, and shall encourage industrial efficiency and competitiveness by limiting the variety of products and processes. Existing standards like ESA's Product Specifications and Standards (PSS) line of documents stated exact details of functions and its quality, together with the means required to produce the wanted products or services. ECSS standards shall be harmonised with international standards or working practices where these have been, or are in the course of being, generally adopted by the European space industry. One of the key element of ECSS is represented by the documentation architecture, which is designed to help the organisation and retrieval of information within the ECSS standards system.
The documentation (ECSS documents as Standards, Handbooks and Technical Memoranda) is basically organized in four main branches that are listed below:

- Space engineering
- Space project management
- Space product assurance
- Space sustainability

The branches are in turn decomposed in several disciplines and domains, as reported in figure 2.10.

The purpose of a space project [92] is to deliver to a customer (and subsequently support or operate if required) a system which includes one or more elements intended for operation in outer space. The activities carried out by the system supplier are conveniently and conventionally categorised into five domains, briefly reported in the following list:

- **Project management**, responsible for achievement of the totality of the project objectives, and specifically for organisation of the project, and its timely and cost-effective execution.
- **Engineering**, responsible for definition of the system, verification that the customer's technical requirements are achieved, and compliance with the applicable project constraints.
- **Production**, responsible for manufacture, assembly and integration of the system, in accordance with the design defined by engineering.
- **Operations**, responsible for exercising and supporting the system in order to achieve the customer's objectives during the operational phases (note; operations may be carried out by the customer, by the supplier or a third party on the customer's behalf, or by a combination of these)
- **Product assurance**, responsible for the implementation of the quality assurance element of the project and also for certain other specialist activities.

The boundaries between such activities are not always clearly defined and formalized since for example:

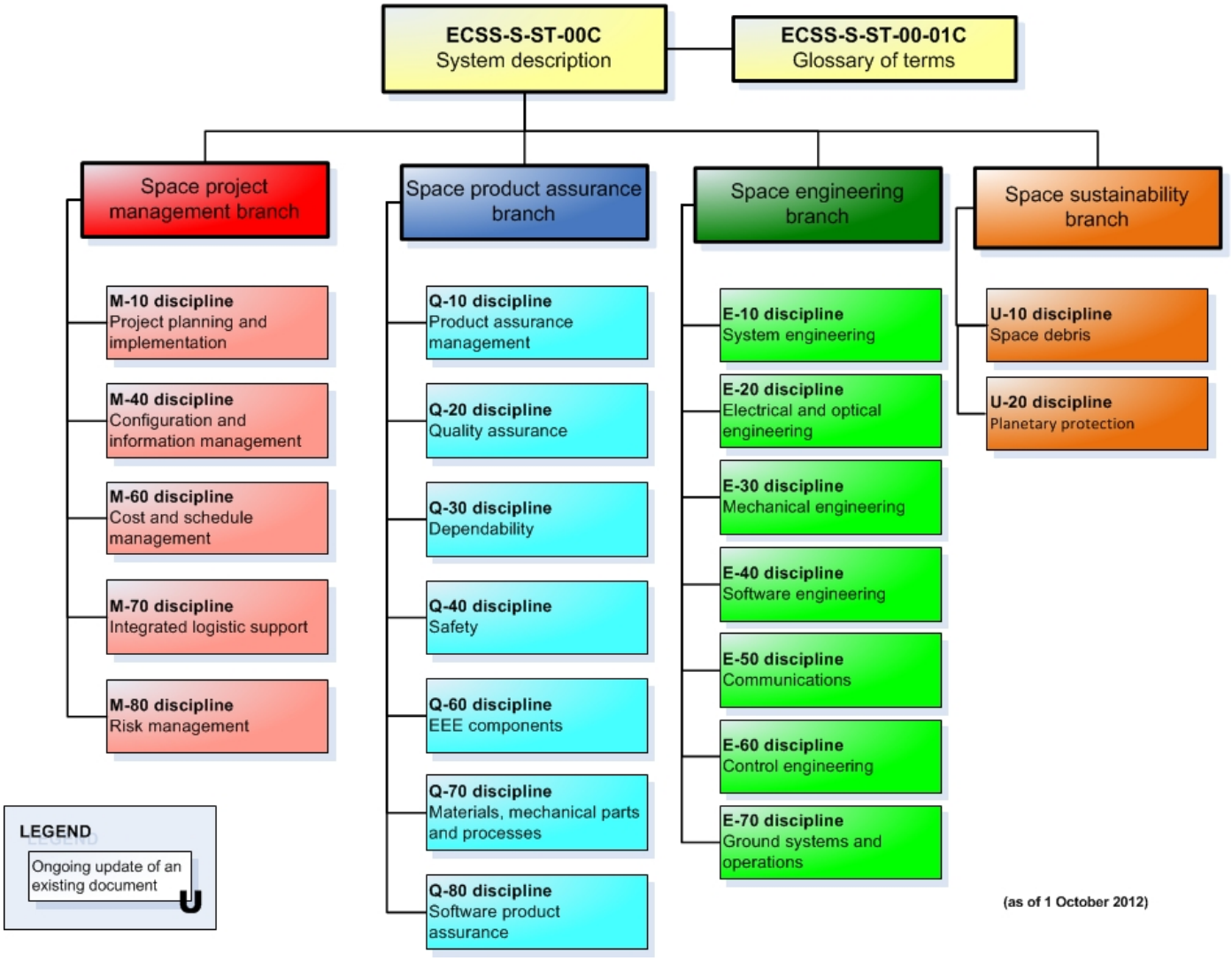

Figure 2.10: ECSS disciplines and domains decomposition [9].

- The engineering, production, operations and product assurance domains each includes an element of management which overlaps with the project management domain proper.
- Production and operations include preparatory and supportive engineering activities, which may also be considered as part of the engineering domain.
- Product assurance includes reliability, availability, maintainability and safety activities, which form an essential part of the design process in the engineering domain.

Harmonization between the three branches of the ECSS system - Management, Product Assurance and Engineering - was initially the activity of a coordination group including the Secretariat and the Technical Panel Chairman. ECSS standards are publicly available documents agreed as a result of consultation and coordination with space agencies in Europe and with industry, and are designed to secure acceptance by users and customers.

Participants in the ECSS incorporate participating member agencies and the European Space Agency (ESA), industry and associates. Associates are those governmental and scientific organizations desiring a formal connection with the ECSS, through which they can monitor the development process of technical documentation and contribute to the ECSS System.

Chapter 3

Model Based System Engineering Methodology

3.1 Introduction

Nowadays the Model Based System Engineering (MBSE) philosophy has started to play an important role for the definition of system model characteristics. The increasing number of variables involved as also the presence of stakeholders often coming from different backgrounds make very difficult to proper manage a complex product. MBSE with respect to the traditional approach provides the basis for a rational organization of work. Some of the features that contribute to make MBSE one of the most spreading modeling philosophies are introduced in the following sections. A particularly comprehensive and clear definition of MBSE is reported in the following lines: "Model-based systems engineering (MBSE) is the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing through-out development and later life cycle phases" and it was available within [10]. One of the main important concept related to the MBSE approach is represented by the term *Architecting*. This definition is strictly related to the process that drives the identification of certain design solutions starting from system objectives. This process is characterized by the analysis of the objects and their relationships for the investigation of the better configuration for the system under evaluation. The main objective is represented by the generation of a balanced architecture where all the elements are harmoniously connected as much as possible. During this phase the system engineering work is also affected by the presence of policies, principles, procedures, budgets, reviews and other activities. Under these conditions the system design process can be characterized by the appearance of omissions, misinterpretations and inconsistencies that later in the development phases can be the sources for a wide range of problems. The main target of MBSE methodology is the reduction of such problems that can considerably affect the system performances or delay foreseen time to market. The generation of a system model in a structured form with a well-defined modelling formalism is one of the most challenging features of the current research topic. In this way it is possible to follow the design during the development process in a more structured manner with respect to the traditional approach as the project matures.
A model based approach shows also the characteristics for a seamlessly integration with object-oriented infrastructures and methodologies. Object-oriented philosophy is currently evaluated for a deeper integration within simulation environments as can be seen for example in [86]. The benefits that can be achieved are reflected in a more effective management of the overall lifecycle of a system.
Traditionally large projects have employed a document-based (also known as document-centric) systems engineering approach. All the information related to the system design and the data exchanged are mainly managed through documents. The generation of textual specifications and design documents characterizes the process of information exchange between all the stakeholders that are involved within the project (customers, users, developers and testers). This approach often lead to a time consuming activities that are not directly related to the project itself, since documents generation, consistencies check and produced drawings validation cover a large amount of time. This approach has deeply influenced the system engineering activity of the last years but when the system begins to increase its complexity this methodology becomes difficult to control and manage. For example requirements traceability becomes even more

challenging when the development process proceeds. This approach of document-based exchange of information and specifications is difficult to update and often results in a poor synchronization between the involved resources. These and other problems can result in an ineffective product development and potential quality issues come out during integration and testing activities. In the worst scenario system faults are discovered once the product has been delivered to the customer. Model-based approach has shown the capability to reduce these problems with an improved management methodology. A mathematical formalism of this methodology has been introduced in the 1993 and electrical and mechanical domains were the first to be characterized. This standard practise has started to spread over other disciplines, showing the benefits of a better structured approach.
System model generally is defined with the support of a modeling tool and all the information is gathered within a model repository and includes data related to specifications, design, analysis and verification. System model is traditionally created from a document-centric vision and all the information are exchanged through documents that often are not well synchronized with the data available for a particular design phase. This situation leads generally to a difficult management of all the information above all when the product complexity increases. In this context the SysML language can provide some interesting capabilities for a correct development of system features. Within the MBSE perspective system model can also be characterized by the integration with engineering analysis and simulation with the final aim to provide useful computation functionality. The other fundamental element of MBSE paradigm is represented by the model repository, highlighting the same importance of the system model itself. This element allow to proper store all the diagrams and information associated to the system model, reporting all the data involved up to a particular phase of system development. In this way it is possible to generate the documentation directly from the model, reducing the time required for the creation of the proper report as in the traditional design methodology. In this case in fact the modeling environment is not directly linked to the tool used for the generation of report and documents. MBSE methodology is instead centred on the formalization of all the system model information within the same modeling tool, allowing for an automatic or semiautomatic generation of reports. In this way the same elements on different diagrams represent the same things and the problems related to consistency check are reduced. This approach limits also the definition of wrong objects since the semantic architecture of the modelled system can be implemented only following certain rules and specifications. In this way is possible to guide the characterization of the data introduced. In the same manner it is possible to better control the possible violations of constrains, reporting the elements affected and providing the instruments for the correction of such situations. The potential benefits, current issues and open points are available from different research initiatives and an interesting description can be found in [70].
The current transition towards a model-based approach is animated by different reasons and a few of them are summarized in the following list.

- Enhancement of communications between all involved stakeholders
- Reduction of development risk
- Quality improvement
- Productivity increase
- Enhancement of the knowledge transfer

Clear definitions of Method and Model term may be useful for the analysis provided in the following sections. Model term represents one or more concepts that are used for the description and evaluation of something in the physical world. Generally the model is an abstract definition referred to a certain domain of interest and does not contain all the required details for the description of the whole system. Models can be generated within different contexts depending on the particular needs for a certain situation. Graphical, mathematical or logical models are all different manners to represent the same system under various perspectives. Also a physical prototype represents a particular form of model that allows representing some particular aspect of the product under evaluation. The model taxonomy of this work follows the definition

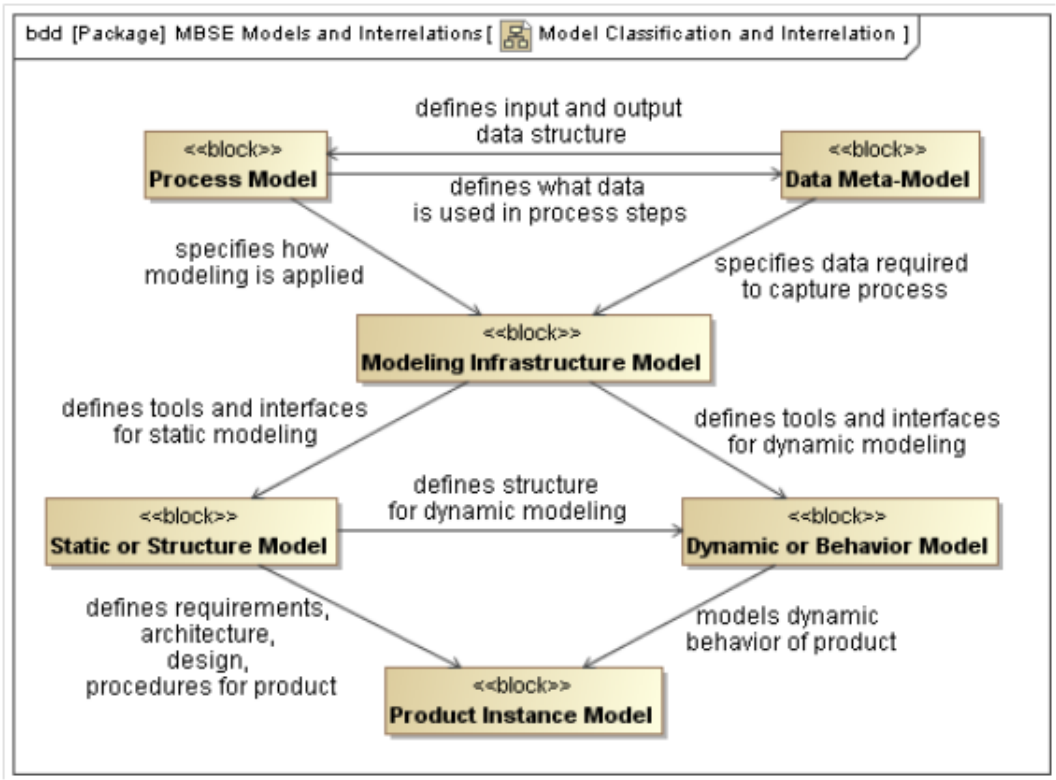

Figure 3.1: Relationships between different kinds of models [11].

introduced by Eisenmann, Miro and De Koning [11] and formally expressed in figure 3.1 where the main six model objects are reported.

The term Method refers generally to a group of activities, techniques and conventions that are used to define one or more processes through the support of tools. This element is fundamental to organize the workflow and to proper define the data exchanges between the stakeholders involved within a certain project. The main objectives of system modeling can be summarized in the following ones.

- Characterization of an existing system
- Specification and design of a new or modified system
- Evaluation of system features/performances
- Training of the users/stakeholders on how to operate and maintain the system

One important distinction must be introduced when considering the model and design terms. A model is defined on the basis of its intended purposes, considering a certain context of applicability. Design refers instead to how well a certain system solution is capable to satisfy customer requirements. The same physical element can be represented with different models on the basis of a certain particular needs. The scope of the model affects significantly the level of resources employed for its implementation and can be formalized through the definition of model breadth, depth and fidelity. Model breadth represents how many elements are needed in the definition of system model for a certain level of implementation. Model depth refers instead to the hierarchical depth of the considered objects. For example the model depth increases as the project proceeds through the development. The same thing is represented on different levels as the object complexity increases as the project becomes more detailed. Finally model fidelity is related to the capability to generate responses that are equal to real results as much as possible.
The consistency of model constraints can be checked through the use of different approaches supported by various instruments, ensuring the correctness of implemented elements. For example object constraint language (OCL) is one of the ways used to formalize the relationships between some system parameters, allowing for a better control of potential requirements violation.

Model definition is a process that requires particular attention when people with different domain-specific education have to interface with each other on the same framework. It is important in this sense to understand how the model is understandable since information not directly needed by the single stakeholder is all presented together. Information overload is in this case one of the problems that may arise during the process of data exchange and visualization. Modeling tools can offer different functionality for the management of such information since all data are formalized following a particular pattern. In this manner the information can be filtered on the basis of the specific needs of the related stakeholder. For example thermal engineers can filter and manage only the information directly related to their thermal domain environment. Model based approach offers also interesting instruments for the investigation of design quality. The availability of formalized data within the system model allows building specific metrics regarding design features for example and these one can be used to evaluate design performance and satisfaction level for the implemented requirements. Different techniques can also be used to monitor the progresses of project development once the system model is defined in the proper way (for example properties managed through standard technical performance measurement - TPM).
In the same manner the progresses and development efforts required to reach a certain degree of completeness of the system project can be monitored, ensuring a better control of the available resources. An estimation of the efforts and costs to complete the design can be foreseen with the use of proper model of investigation in the context of model based approach. This last feature represents surely one of the main advantages with respect to the traditional design methodology.
In order to better explain the concepts that will be introduced in the following sections should be useful to properly define the terminology that will be used. A brief explanation of the terms used will help to avoid the possible misunderstandings about the topics and features that will faced. The definitions used for the explanation of the current concepts come from the work [12], where a clear distinction among words often used as synonymous is provided. The word *methodology* is often expressed as synonymous of the word process while they should refer to different concepts. To better understand the differences between such two words the definitions of *process*, *method*, *tool* and *environment* are considered:

- A **Process** is a logical sequence of tasks performed to achieve a particular objective. Such term basically defines “WHAT” is to be done, independently from the way such tasks are done.

- A **Method** includes the techniques that are used to perform a certain task, defining “HOW” each task must be done. The word method can be alternatively interchanged with the term technique, practise and procedure. The process tasks are basically performed using methods and such pattern is repeated for different detail levels. In particular each method can also be seen as a process itself since the “HOW” of one level becomes the “WHAT” for the next lower level.

- A **Tool** is an instrument that is applied in the context of a particular method to improve the efficiency of a specific task. The application of a specific tool is often realized through somebody with proper skills and training. Referring to the previous definition a tool can be considered as the element that enhances both the “HOW” and the “WHAT”. Computer Aided Engineering (CAE) tools fall within such class since they are conceived mainly to support the design and analysis phases for system development.

- An **Environment** represents the surroundings, the external elements, conditions, or factors that affect the actions and/or responses of an object, individual person or group. The cited conditions can be represented by social, cultural, personal, functional, organizational or physical events.

Once these concepts have been clarified through the proper definition of the related terminology the word methodology can be better understood. In particular a methodology can be defined as a collection of related processes, methods and tools that are conceived and integrated to approach a certain class of problems that share some common element (as expressed in [13]). The main focus of a project environment is to provide the proper support for the integration and application of tools and methods used in the related project. The relationships between all the just introduced concepts can be graphically represented in figure 3.2, where the correlations with technology and people are also reported [12]. Technology capabilities and

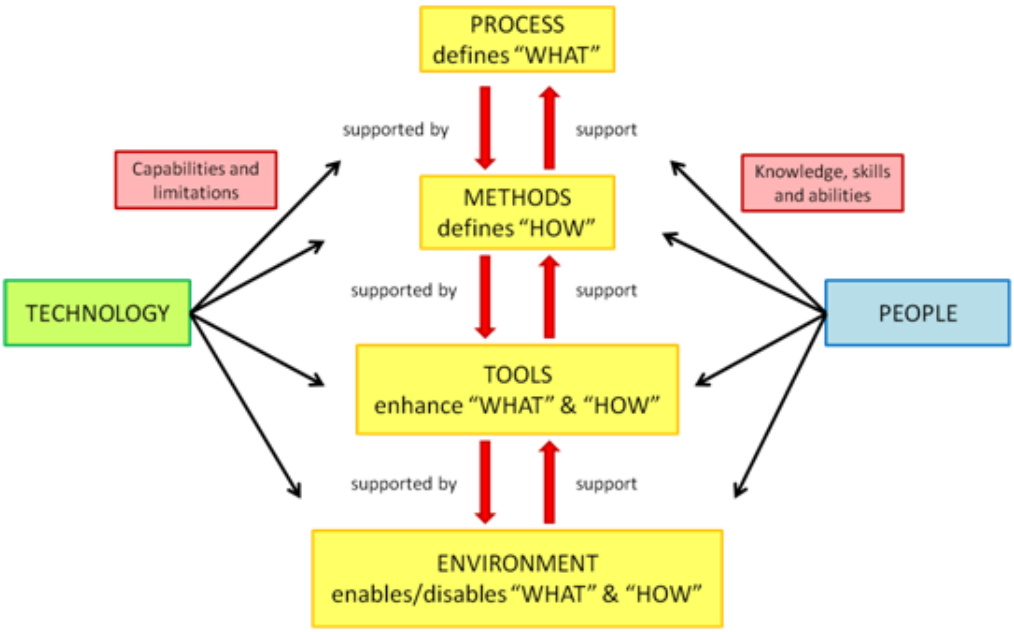

Figure 3.2: Process, Methods, Tools and Environment elements and relationships with technology and people.

limitations must be well understood before the definition of a methodology and related infrastructure. In fact the correct development of a project will be affected in the way the technology is exploited since it can help or slowdown system engineering efforts. The definition of methodology infrastructure requires a balanced partitioning of Process, Methods, Tools, and Environment, considering also the knowledge, skills and abilities (KSA) of the people involved.

3.2 INCOSE initiative

The International Council of System Engineering (INCOSE) represents the organization that covers a key-role in the definition of MBSE ontology and its spreading all over different engineering fields both in academic and industry. Regular workshops and conferences are organized by INCOSE to support the integration of such methodologies within the current system engineering strategies. A well-defined MBSE roadmap has been identified to schedule the main objectives and improvements that might be reached in future developments. The desired maturity levels with respect to the temporal evolution are reported in figure 3.3 as presented in [14].

The conceptual pattern refers to the areas reported in the lower-right corner and they are also briefly repeated in the following list:

- Planning and support
- Research
- Standards development
- Processes, practices and methods
- Tools and technology enhancements
- Outreach, training and education

The main topic of the current work can be related with the concepts presented in the just introduced diagram. In particular the capabilities I refer to are both represented by design optimization across broad trade space and cross domain effects based analysis.

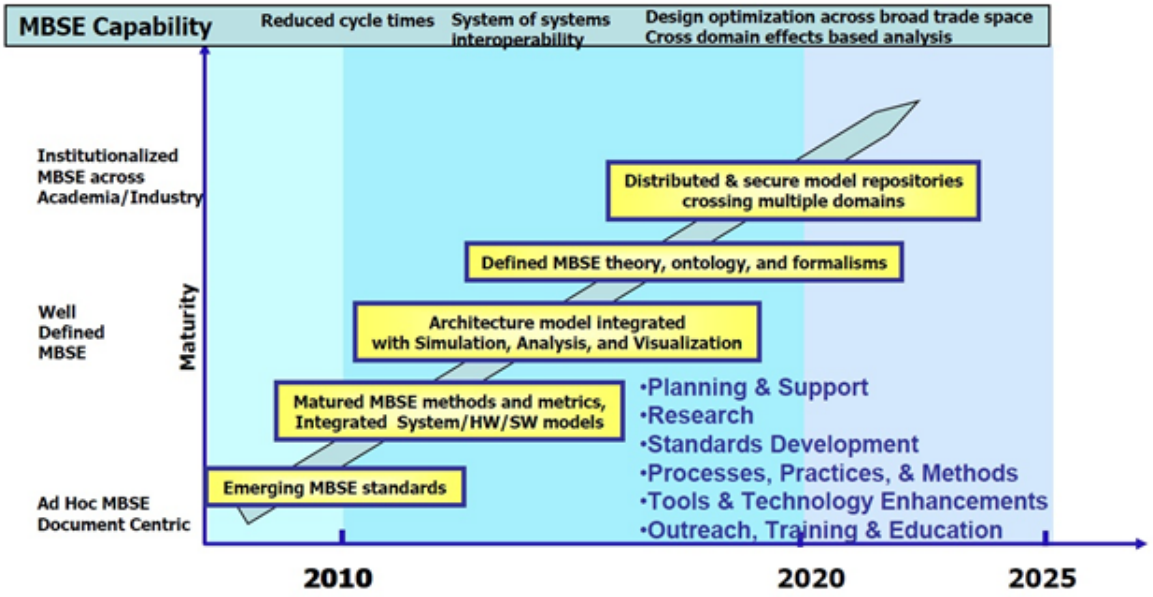

Figure 3.3: INCOSE MBSE Roadmap [14].

The primary efforts are addressed towards the improvement of MBSE architecture and their maturity level. In particular the main direction is represented by the passage from the emerging MBSE standards to distributed and secure model repositories crossing multiple domains. The extension of maturity level is concurrently characterized also by the improvement of MBSE capabilities. In this context the reduced cycle times turns into system of systems interoperability and finally introducing design optimization across broad trade space. In this utopian vision the MBSE paradigm provides a clear environment for analyses based on cross domain effects. Interesting challenge teams and research groups have been involved within the INCOSE initiative and space applications have been approached with such methodology. The INCOSE Space System Working Group (SSWG) is one of such teams and their reference case to assess the benefits of such modeling technique has been identified with the FireSat mission, available from the literature (Space Mission Analysis and Design [15]).

3.2.1 System modeling language - SysML

Currently one of the most widespread modeling languages that has been used for the definition of system characteristics from different viewpoints is represented by System Modeling Language (SysML). In particular this language is now drawing the attention in the context of system engineering due to the well suited advantage to model a high number of system features, starting from the topological ones but also covering other fields as the operational and functional characteristics for example. Its capability to represents the main features in a flexible way and covering different domain- specific modeling techniques is one of the most interesting key-role. Currently different research groups are involved in the assessment of the feasibility of aerospace system modeling. The main aim is the evaluation of the potential benefits and drawbacks related to the modeling of complex system (such as product related to space application), where a wide range of people with different skills and backgrounds are involved on the same project. In particular study research topics are addressed towards the understanding of the actual scalability of SysML to system with a high level of details and the possible integration of such language with automated code generation. This last feature is directly related to the possibility to run simulations starting from the “representative” model definition. With this last term we identify the model that contains all the information related to the model characteristics but it does not contain codes or similar runnable simulation model. In this way the evaluation of system performances can be realized already in the early phase of the project. In this case one of the main challenging topics is represented by the integration between the representative system model and external (or SysML embedded) simulation solver (as for example external simulation proprietary tools). Currently an increasing number of commercial tools offer the capability to support and

develop MBSE concepts within the system project. In particular as SysML/UML tools offer the implementation of simulation capabilities through the installation of proper plug-ins also multidisciplinary simulation tools offer the functionality to implement some of the MBSE project methodologies. In the first case SysML tools allow to build simulation codes starting from the available information provided through particular classes of diagrams. For example the parametric diagrams are defined within SysML environment and they are mainly used with this purpose. They allow formalizing the physical and constraint relationships between the model classes and objects introduced within the project. Numerical values and parameters can be related to the object in a more effective way with respect to the traditional approach, reducing the possibility to introduce consistence errors between the modeled elements. This information can then be used to build simulation codes that are sent for example to external solvers. The results coming from these simulations can then be post-processed in the same modeling environment with the proper instruments. In this way is possible to manage and check the possible inconsistency between the elements involved within the product development.
For this reasons some projects are currently evaluating the SysML for the architecture design, simulation and visualization as reported in [83].
From the previous consideration SysML seems to be an interesting language for the modeling of system features from different point of view, above all considering the different disciplines that have to deeply interact during the development. One of the main benefit related to SysML modeling tools are represented by the flexibility to manage different aspects of the same project. This modeling approach is addressed to system engineering that have to monitor and check a wide range of variables and parameters during the project. In this case the system engineers have to learn a new modeling instrument for their space application purposes. This last aspect seems to be one of the drawbacks related to the use of SysML language. This language is however designed to unify the diverse modeling language currently used by system engineers as Unified Modeling Language (UML) is conceived to standardize the modeling languages used by software engineers. As previously introduced SysML allows supporting the specifications, analysis, designs, verifications and validations of a wide range of complex systems. The diagrams used for such a purpose are represented by the Block Definition diagram, Internal Block diagram, Package diagram, Parametric diagram, Requirements diagram, Activity diagram and Use Case diagram. Nowadays different commercial software-houses offer SysML plug-in as complementary elements for their UML software suites.
The SysML specification includes the definition of the previously set of diagrams that allow to manage all the system information in a consistent way. Each diagram is related to a particular aspect of the system architecture and offers a wide range of features for the particular element to be modeled. These diagrams can be associated to four main groups which are often denoted as the four pillars of SysML language. A conceptual overview of these four pillars is represented in figure 3.4.

SysML Block Definition Diagrams (BDD)

This class of diagrams is used to define the features of a block and any other relationships between blocks such as associations, generalizations and dependencies, characterizing properties, operations and attributes. This kind of diagrams is generally used to model the system hierarchy or a system classification tree. They are used to clearly define structural composition, interconnection and classification of the involved technologies. Function-based representations are also integrated and allow to model state-based behavior of the system. In particular this diagram is used to represent structural elements (also defined as blocks), their relationships, compositions and classifications. SysML BDD is derived from UML class diagram with some modifications.

SysML Internal Block Diagram (IBD)

SysML Internal Block Diagram (IBD) is typically used to model the restrictions and extensions that characterize the represented element. An IBD captures the internal structure of a Block in terms of properties and connections among the properties. In this case the ports, the connectors and the linked parts are represented with the final aim to highlight how the objects are internally defined. SysML IBD has been derived

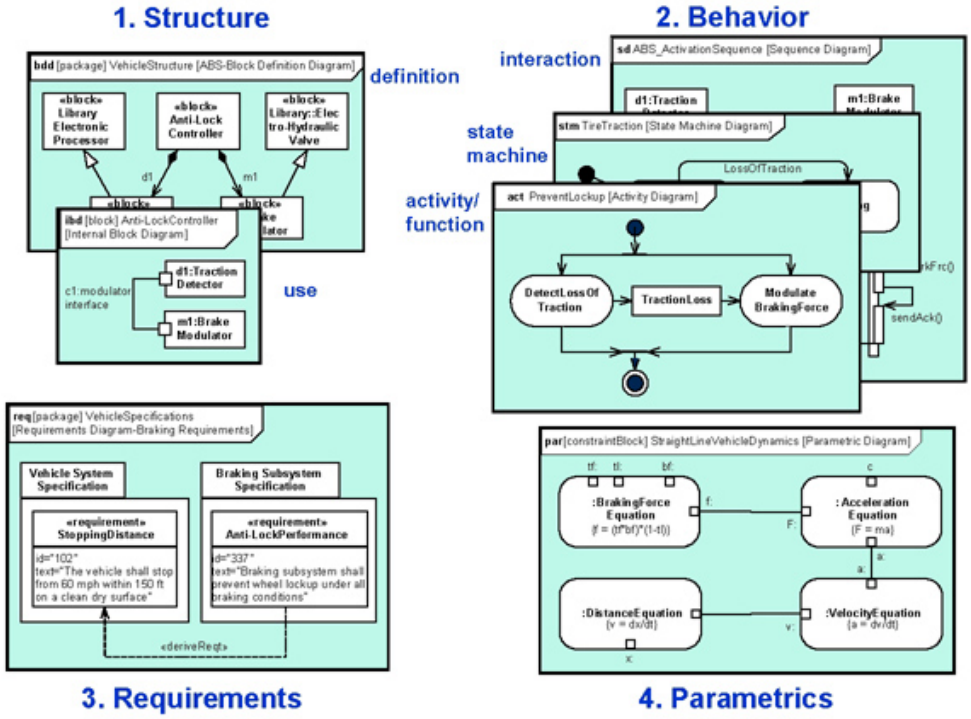

Figure 3.4: Pillars of SysML language [102].

from UML composite structure diagram with little modifications.

SysML Package Diagram

Another class of diagrams is represented by the SysML package diagrams. They are typically introduced to organize models by partitioning model elements into groups and establishing also the dependencies between other packages or model elements. The project can be effectively organized in a more suited way with the package elements since also the view object are organized within such representation. UML language also implements the same kind of diagrams with no differences.

SysML Parametric Diagram

SysML Parametric Diagram can be considered as a special case of the IBD class. They are quite similar with the only difference represented by the fact that the connectors allowed are the binding connectors. Such diagrams are mainly used to model the constraints that affect the properties of a particular block. They will contain both constraint properties and constraint parameters, defining the relationships that bound certain parameters to other one. In this way it is possible to model physical relationships, constraints and similar associations between the parameters modeled. This kind of diagram is generally used to build trade-off analysis for the configurations modeled in the same project. A constraint block can be used for example to define an objective function to compare all the available and alternative solutions. This kind of diagram is not present in UML modeling environment.

SysML Requirements Diagram

SysML Requirements Diagram is one of the most interesting elements introduced within the SysML formalism since UML does not include such representation types. Text-based requirements can be stored properly and it is also possible to clearly define relationships with other requirements, design objects and also test cases. In this manner it is possible to ensure a well-organized traceability between the various elements involved in the design process.

SysML Activity Diagram

SysML Activity Diagram has been derived from UML with little modifications. This type of representation is used mainly to model the behavior of the system with respect to the input and output flows that characterize the interconnections between objects. In particular it is possible to order actions that interact between each other on the availability of inputs and outputs, defining how the actions themselves transform these ones.

SysML Use Case Diagram

SysML Use Case Diagram is used to represent the functionalities that the system is able to accomplish, considering in particular how the involved entities are used or managed by external users or other elements. The main aim of this visualization is to clearly define the relationships of the involved entities in reaching some targets. This type of diagram is also present within the UML language specification.

SysML Sequence Diagram

SysML Sequence Diagram has a corresponding representation within the UML language. This type of diagram equivalently has been conceived to report the behavior of the system of interest as a temporal sequence of the information exchanged (physical quantities, electrical signals, messages, etc. for example). This kind of representation is quite similar to the one considered in the activity diagrams but in this case the focus is on the temporal evolution of the involved actions rather than the entity involved within the process.

SysML State Machine Diagram

SysML State Machine Diagram is generally used to model the behavior of a system in terms of the transitions between the different stages that characterize how the system model responds to commands or external inputs. These events can trigger the transition from one stage to another for the system under evaluation and this aspect is captured within state machine diagrams.
The diagram type chosen for the modeling of a certain system feature constrains the elements that can be modeled within such diagram and this characteristic reduces the possibility to introduce errors within the modeled context.
SysML represents a valid alternative for the management of complex system from its early development phases and its main use is related to the support of activities in the context of MBSE methodology. SysML does not impose a specific method to model system information starting from the requirements. The method chosen is strictly related to the industry knowledge and development process pattern which determine what activities are performed first and the artifact that have to be delivered before pass to the following action. One method is represented by the decomposition of system function from the requirements specification. The identified functions are then allocated to components and then system performances are evaluated before starting another decomposition for a more detailed level. Alternatively the use case driven approach starts from the scenarios that the system has to face. The functionalities that the system must show are derived from the operational scenarios and all the functions are then derived and allocated to the various elements. The interactions among parts are then investigated and better defined before proceeds to the next phases. In both case the illustrated processes are iteratively performed until a satisfactory design has been obtained (consistently with the customer needs). The two methods can produce different diagrams and information in various manners to represent system design but in both cases SysML can be used to support and formalize the modeling activities.
SysML language can be used iteratively to proper obtain the final design and some of the main involved activities are summarized conceptually in the following list.

- Capture, analyze and formalize system requirements

- Define and develop one or more design solutions to satisfy customer needs
- Perform engineering and trade-off surveys to investigate and identify/select a valid architecture
- Specify and allocate requirements to components, ensuring traceability to system requirements
- Verify that the design satisfy the requirements performing system-level test cases

One of the main advantages of SysML semantics is the development of an integrated and consistent model where the model objects defined on one diagram can be related to model objects on other diagram (representing however the same entity), avoiding the need to redefine the same element just created and automatically ensure that the element is consistent with the other representations of itself (on other diagrams for example).

3.2.2 Taxonomy and definitions

The notation used to describe the features of the proposed infrastructure is based on the UML/SysML notation. In this way it has been possible to formalize all the concepts used for the definition of the conceptual framework. The related notation includes in fact all the elements needed to clearly define the objects and conceptual data. The definition of model and meta-model terms is strictly related to the following conceptual activities and a first explanation of such concepts is provided in the following lines. In this context they are used to better describe SysML/UML notation but a more extended description is provided for the definition of the overall conceptual infrastructure (in the following chapters).
A **model** can basically be defined as an abstract and conceptual representation of a system (or generally of any conceivable entity) that emphasizes both the "sub-objects" (or sub-entities) which compose the overall system and the relationships between them and their properties and behaviours.
On the other side a **meta-model** generally defines all the "rules" regarding how a model must be done. It highlights in particular what kind of objects, properties, and relationships the model could envisage, and how properties and objects contain or are associated to others.
A more explanatory definition for the meta-model term is available from [91] and is provided in the following lines for the sake of clarity.

"A view of the real world (i.e. human--oriented concepts) in terms of object types, their characteristics and relations between object types. Knowledge about the real world is expressed in terms of elementary facts, constraints and derivation rules. Different terminology is used for the same concepts within different modelling methodologies. [...] A conceptual model can be seen as a network of object types and relations, further refined by constraints and rules that shall or should be satisfied. Some of these relations are of part of nature and of special interest for determining the user views of the model, i.e. a conceptual model is not just a network of definitions but organized into hierarchical sets of definitions that represent the user views"

Meta-model notation is derived from the SysML one, referring to the Unified Modeling Language(UML) specification, the conceptual model is defined using the standard UML class diagram notation. To comprehend the meta-model diagram it is first necessary to describe the typology of existing relations between the different Engineering Data Item (EDI) Classes which compose the model database (EDI classes concepts are detailed in the appendices). Conceptual data are represented by rectangles in the diagrams, and relations are represented by arrows with different ends, depending on their role, which link the rectangles. A conceptual data represents a concept whose instantiation is a specific data item, related to definition of a class. There are four main kind of relation:

- **Association** is a relationship where all object have their own lifecycle and there is no owner. Let's take an example of Teacher and Student. Multiple students can associate with single teacher and single student can associate with multiple teachers but there is no ownership between the objects and both have their own lifecycle. Both can create and delete independently.

- **Aggregation** is a specialize form of Association where all object have their own lifecycle but there is ownership and child object cannot belongs to another parent object. Let's take an example of Department and teacher. A single teacher cannot belong to multiple departments, but if we delete the department teacher object will not destroy. We can think about "has-a" relationship.
  Aggregation differs from ordinary composition in that it does not imply ownership. In composition, when the owning object is destroyed, so are the contained objects. In aggregation, this is not necessarily true. For example, a university owns various departments (e.g., chemistry), and each department has a number of professors. If the university closes, the departments will no longer exist, but the professors in those departments will continue to exist. Therefore, a University can be seen as a composition of departments, whereas departments have an aggregation of professors. In addition, a Professor could work in more than one department, but a department could not be part of more than one university.

- **Composition** is again specialize form of Aggregation and we can call this as a "death" relationship. It is a strong type of Aggregation. Child object does not have their lifecycle and if parent object deletes all child objects will also be deleted. Let's take again an example of relationship between House and rooms. House can contain multiple rooms there is no independent life of room and any room cannot belongs to two different house if we delete the house room will automatically delete. Let's take another example relationship between Questions and options. Single questions can have multiple options and option cannot belong to multiple questions. If we delete questions options will automatically delete.
  Both are ways of designating or grouping items by relationship. In the case of composition, if the links that bind the objects are broken, then all objects are destroyed. In aggregation, it's a looser grouping, and if the links are broken the original objects still exist.

- **Generalization** is not an Association, because it is not defined at instance level, but at class level. It indicates that a concept is a subtype with respect to a second concept, in the sense that the last one comprises the first one. It is indicated with a hollow triangle in the side of the general concept. It is basically used to define the relationship between two classes where one of them inherits the attributes and functions from the more general one. In this way it is possible to group the properties and methods that are common across different classes while the specialized class can include specific attributes and functions.

Multiplicity defines the number of instances of one class which may be linked to one instance of the other class. They indicate the maximum and the minimum allowed value of this number. Relations described above are represented (perhaps even more clearly) in figure 3.5:

3.2.3 SysML tools

Different solutions following SysML specification are currently available both commercial and open-source. A brief list of SysML commercial tools is represented in the following.

- Enterprise Architect + MDG Technology for SysML (vendor: Sparx Systems)
- UModel Enterprise Edition (vendor: Altova)
- MagicDraw + SysML plugin (vendor: No Magic)
- Rational Rhapsody Developer (vendor: IBM)
- Artisan Studio (vendor: Atego)

The following list reports instead some of the open-source SysML modeling tools and plugins, which are typically free to use for personal use and their utilization is regulated by open-source licensing conditions.

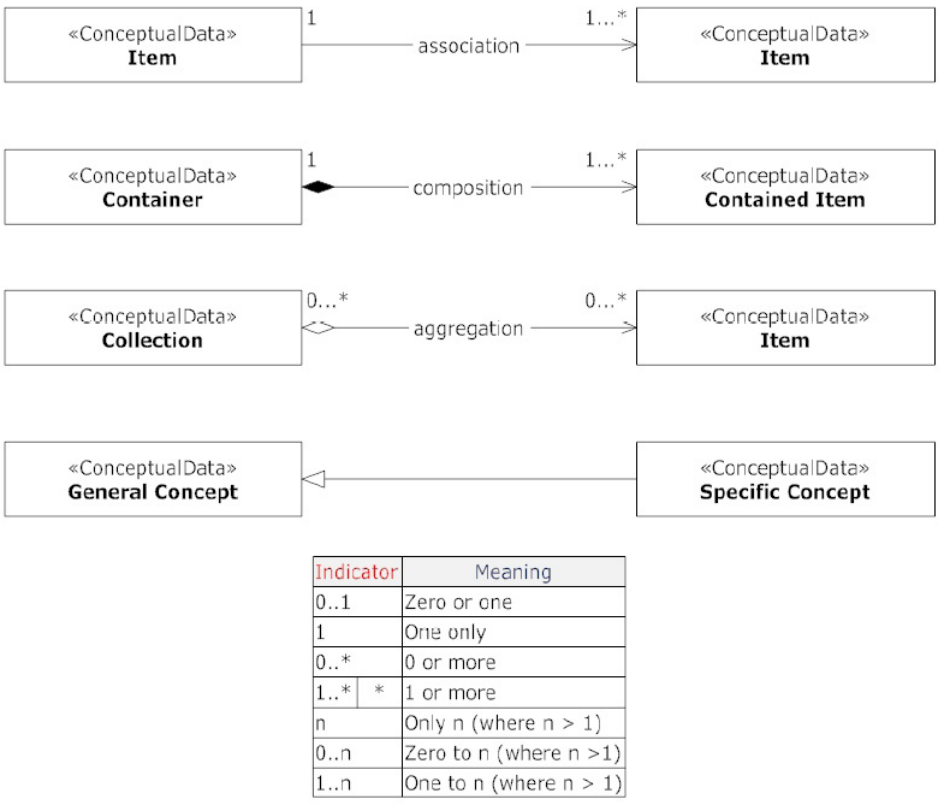

Figure 3.5: Notation for the main relations used to define the object belonging to the overall meta-model.

- Modelio Free Edition + Modelio SysML Designer module (source: Modelio Open Project)
- TOPCASED-SysML (source: TOPCASED Modeling Framework Open Source Project)
- Papyrus for SysML (source: Papyrus Open Source Project)

3.2.4 Semantically-Rigorous System Engineering using SysML and OWL

SysML language is currently one of the most widespread and accepted graphical modeling solution for system engineering. OMG specifications are supporting such modeling language, providing useful help for the management of system engineering activities. OWL language (which stands for Ontology Web Language) is currently widespread as knowledge representation language. The specifications of such language are defined by the W3 industry consensus and the main strength is represented by the logical formalism and general applicability. Currently different research activities are addressed to the integration of both this formal modeling language ([16]). One of the most important features related to system design is strictly related to the definition of logical reasoning formalism. This element can be involved in the requirements tracing process, allowing for a better management of the specifications themselves. The interface compatibility can also be better checked through the formalism defined within this context. Another important aspect is also related to the control of viewpoint consistency between different system perspectives.
The main activity relate to the integration of OWL and SysML languages are represented by the definition of a well based set of OWL ontologies for the formalization of general concept and properties in a hierarchical context. These categories can be represented for example by discipline, application, mission and project. The main idea is represented by the development of OWL ontologies for SysML. These are then used to formalize and capture the object properties with the main aim to allow for SysML to OWL transformation. One of the other interesting features is represented by the extraction and transformation for specialized analysis tools (through a clear formal representation of the exchanged data – Maple and Mathematica for example). Future developments regard mainly the possibility to simplify the profile generation code.

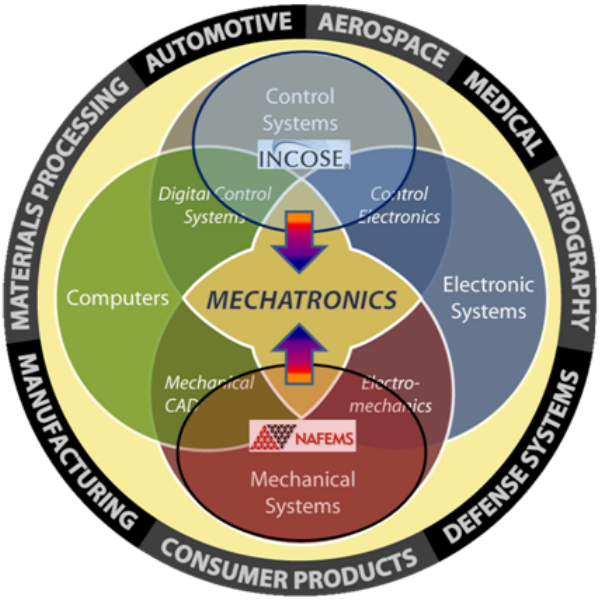

Figure 3.6: Convergence process between INCOSE and NAFEMS [17].

3.2.5 Systems Modeling & Simulation Working Group (SMSWG)

The integration of simulation and analysis capabilities within a model based infrastructure is one of the most challenging and promising research activities. Such interest is also highlighted by the recent creation of a joint working group between the International Council on Systems Engineering (INCOSE) and the International Association of the Engineering Modelling, Analysis and Simulation Community (NAFEMS). Such joint initiative is addressed towards the mutual participation and collaboration for the advancement of engineering simulation and model based systems engineering. One of the main objectives is represented by the promotion of a deeper understanding of the integration strategies for mechanical analysis and simulation within a model based system engineering environment. Nowadays the successful evolution of MBSE methodologies with the supervision of INCOSE has shown the significant opportunities which come out from a stronger cooperation with key engineering disciplines such as software and CAE. More details about such cooperation can be found in the presentation concerning the NAFEMS – INCOSE Collaboration Kick Off during the INCOSE International Workshop 2013 [17]. This recent joint relationships highlights how the integration between a model-based system modeling environment and analysis framework is currently one of the most investigated area of research. The purpose of such initiative can be summarized in figure 3.6where the convergence between the corresponding project is highlighted.

3.3 Collaborative environments

Collaborative Working Environments (CWE) have been conceived to support people work both individually as well as from a cooperative perspective. The main objective of such environments is addressed towards the definition of infrastructures that ideally involve people not directly working within the same geographical place. The capability to enhance the overall effectiveness with respect to a particular project is strictly related to different aspects that range from support facilities to conceptual organization of the work. Application sharing, document management, collaborative workspace or workflow organization are some of the elements that generally make the difference in the development process of a certain product. In the last few years the development of increasingly complex system has lead towards the definition and investigation of a wide range of collaborative architectures and related processes. The main aim of such research activities has been addressed to the identification of the best solution for the management of all the available resources. Different collaborative infrastructures have been conceived with such purpose and they are mainly applied during the preliminary design phases. Such infrastructures become difficult to manage and properly exploit as the development process move towards more detailed steps. One of the

main objectives of the present work is also represented by the investigation of the possible improvements that a MBSE methodology can introduce in the design of space systems during the advanced stages of the project (phases B-C for example). Currently some research efforts are also addressed towards the analysis of the possible alternative solutions that can be chosen for the integration of a distributed environment and web-services capabilities.
Some of the most interesting research activities performed few years ago have led to the development of a collaborative environment which is known as Concurrent Design Facility (CDF). CDF has been mainly conceived for the preliminary phases of space systems and it is basically a state-of-the-art facility equipped with a network of computers, multimedia devices and software tools, which allows a team of experts from several disciplines to apply the concurrent engineering methods to the design of future space missions [18]. The main focus of such infrastructure can be identified with the capability to perform the assessment studies of future missions with fast and effective interaction of all disciplines involved, ensuring consistent and high-quality results in a much shorter time. The activities that can be performed within such environment can be summarized in the following ones: conceptual design, mission trade-offs, reviews of industrial phase A studies, scientific requirements definition and consolidation, options evaluation, new technology validation, anomaly investigation, education and training [19].
The Concurrent Design Facility was established at ESTEC in November 1998 within the framework of the General Studies Programme and has been directly involved on different European scientific missions.
Collaborative environments basically built on the basis of the CDF architecture can be found within different scientific organization or industrial companies. They are mainly used in the field of space systems to support the preliminary development phases. Such environments are currently not conceived to be used in the advanced phases of the project where a collaborative process is strictly affected by each company expertise or project settings. CDF shows some interesting advantages with respect to the traditional approach in the management of a collaborative workflow. In this case a more effective improvement may be achieved with an extension of the related philosophy also to the more detailed phases of the design. The related concepts can in fact also applied in the case of more advanced phases but such integration requires the redefinition of the overall infrastructure. The main ideas are still valid but the constraints linked to the resources involved during the overall process need to be properly managed. CDFs are generally represented by open-space workplace where around twenty persons work together, involving basically different specialists coming from various domains. The same approach is not practically applicable in the more advanced phases and the overall architecture must be rethought. In this case a distributed environment seems to show some promising features with respect to previous concept of a collaborative environment located in the same place, corresponding also to the same room in the case of CDF.
Working environments based on distributed approach can enhance the multidisciplinary integration across the advanced phases of the project, easing the coordination and collaboration between different engineering teams often located in different places (not necessarily in the same room). Such situation generally characterizes the advanced phases of system design where groups of different domains start to grow with respect the preliminary phases. During the later phases the number of people and resources involved becomes difficult to manage through an infrastructure similar to CDF and alternative solutions are then considered. New model-based methodologies have been developed to manage the next-generation complex systems and they are mainly conceived in the context of collaborative environments. Interesting results are provided by different research initiatives and [103], [104] and [105] are examples of these projects (a conceptual representation of the environment proposed in [104] is provided in figure 3.7).

One of the main important aspects in the context of Collaborative Environments is represented by the data exchange and such element must be taken into account with particular attention. The correct definition of such process makes the difference between an effective and an ineffective collaborative workplace. The right definition of data exchange process must be built from a conceptual model of data classes and their relationships. A more detailed description about data model will be introduced in the following.

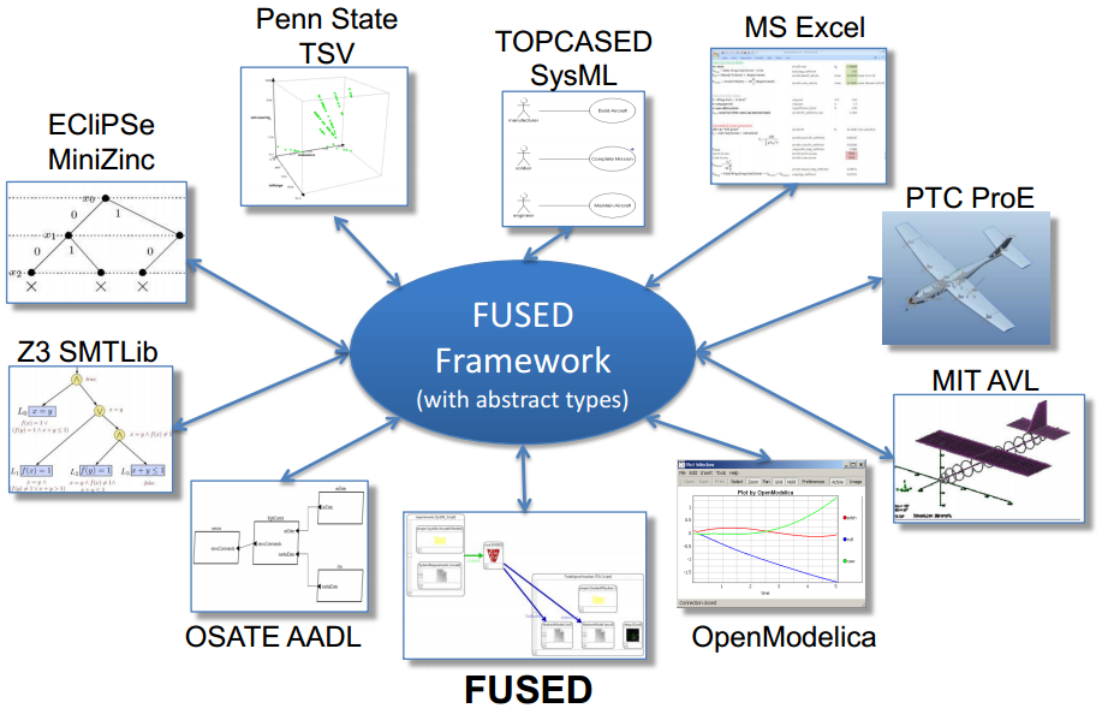

Figure 3.7: FUSED Framework: control and data flows between models [105].

3.4 Examples of MBSE initiatives and Collaborative Engineering environments

Currently there are different research initiatives focused on the evaluation of MBSE methodology as a useful infrastructure for the management of complex systems. In particular such activities have mainly involved companies and organizations that work on aerospace systems which have started first to investigate the potential benefits of such approach with respect to the traditional one. An interesting survey on the advancing of System Engineering methodologies and on the early phases of integration of MBSE methods is provided in [88]. In particular the main focus of such activity is represented by experience matured through the Systems Engineering Advancement (SEA) Project. The SEA Project developed products, services, and training to support managers and practitioners throughout the entire system life-cycle. In particular the main efforts were addressed towards the investigation for the possible improvements of the following functions and activities:

- Systems architecture
- Requirements management
- Interface definition
- Technical resource management
- System design and analysis
- System verification and validation
- Risk management
- Technical peer reviews
- Design process management
- Systems engineering task management

This analysis shows how the need for a well-defined methodology for the management of such aspects is of great interests several years ago.
The design and analysis of complex aerospace products have been characterized earlier by such innovative methodologies but the same concepts can likewise be applied in other engineering domains (mechanical, biomedical, etc.). The basic concepts and the related formal infrastructure can in fact be exploited to manage other kind of systems.
In the following sections some examples about the assessment of MBSE methodology are reported. In particular they refer mainly to the the application of such methods to space systems.

3.4.1 Responsive Engineering

The actual development processes related to the implementation of the design engineering tools are all animated by the same concepts. Some of the main conceptual elements that will affect the future development are represented by parallelization of tasks, building block modular approach, standardization of interfaces, standardization of requirements and acceptance of higher risk and possibly of lower quality. Some other interesting aspects are the integration of design tool, test by simulator, fast AIT, avoidance of just in time procurement and finally a responsive procurement method [20].
Interesting activities are represented in this direction by different research evaluation project. Other interesting initiatives are directly involved within the evaluation of space system project as for example the NASA RSDO. The Rapid Spacecraft Development Office (RSDO) is responsible for example for the management and direction of a dynamic and versatile program directing the definition, competition, and acquisition of multiple fixed-price Indefinite Delivery/Indefinite Quantity (IDIQ) contracts. These contracts offer NASA and other Federal Government Agencies fast and flexible procurement of spacecraft and spacecraft components for future missions.

3.4.2 ESA-ESTEC initiative

Concurrent Engineering activities at ESA can be basically identified with the development of the Open Concurrent Design Tool (OCDT) initiative. The main efforts are addressed in the application of such methodologies as support instrument for feasibility studies. The other primary activities are strictly related to the formalization and standardization of the conceptual data model. An upgrade of collaborative environment is under development and highly secure connection for external resources as also the authentication procedures are under implementation. The same environment has been conceived for future distributed design sessions, review support procedures and anomalies investigations. Such activities are all considered as new applications of Concurrent Design Facility (CDF).
The OCDT architecture can be briefly represented in the following overview (figure 3.8).

A central persistent data repository is used to store and synchronize the information related to the system model while HTTP(S) protocol allows consistent connections with ConCORDE framework (Excel based). The single domain uses the domain specific tools while ConCORDE ensures data exchange with the other disciplines. It is basically an Add-In on top of Microsoft Excel 2010 with the main aim to reduce as much as possible the learning curve (since Excel environment is one of the most used tool in some engineering fields). In this way the transition from existing CDF IDM architecture is not so critical and at the same time Excel shows some flexible features. Excel provides in fact the capability to rapid create interfaces with any kind of external tool as the generation or modification of worksheet for computation purposes. ConCORDE provides reference data library (parameter types, units and scales, rules, etc…), engineering model catalogue (Options, Element Definition, etc…) or other utilities as support to design activities. IDM and OCDT architecture show some differences. The main differences are related to the information exchange process, data ownership, decomposition level and options management. It is interesting to note above all the management of system options since this topic will be deeper investigated in the current work. IDM approach supports an option specific architecture basically builds by copy and paste. In this case there is one workbook for option. OCDT approach instead is based on different philosophy. The options are defined singularly and the various architectures are generated through an automated process staring from the re-

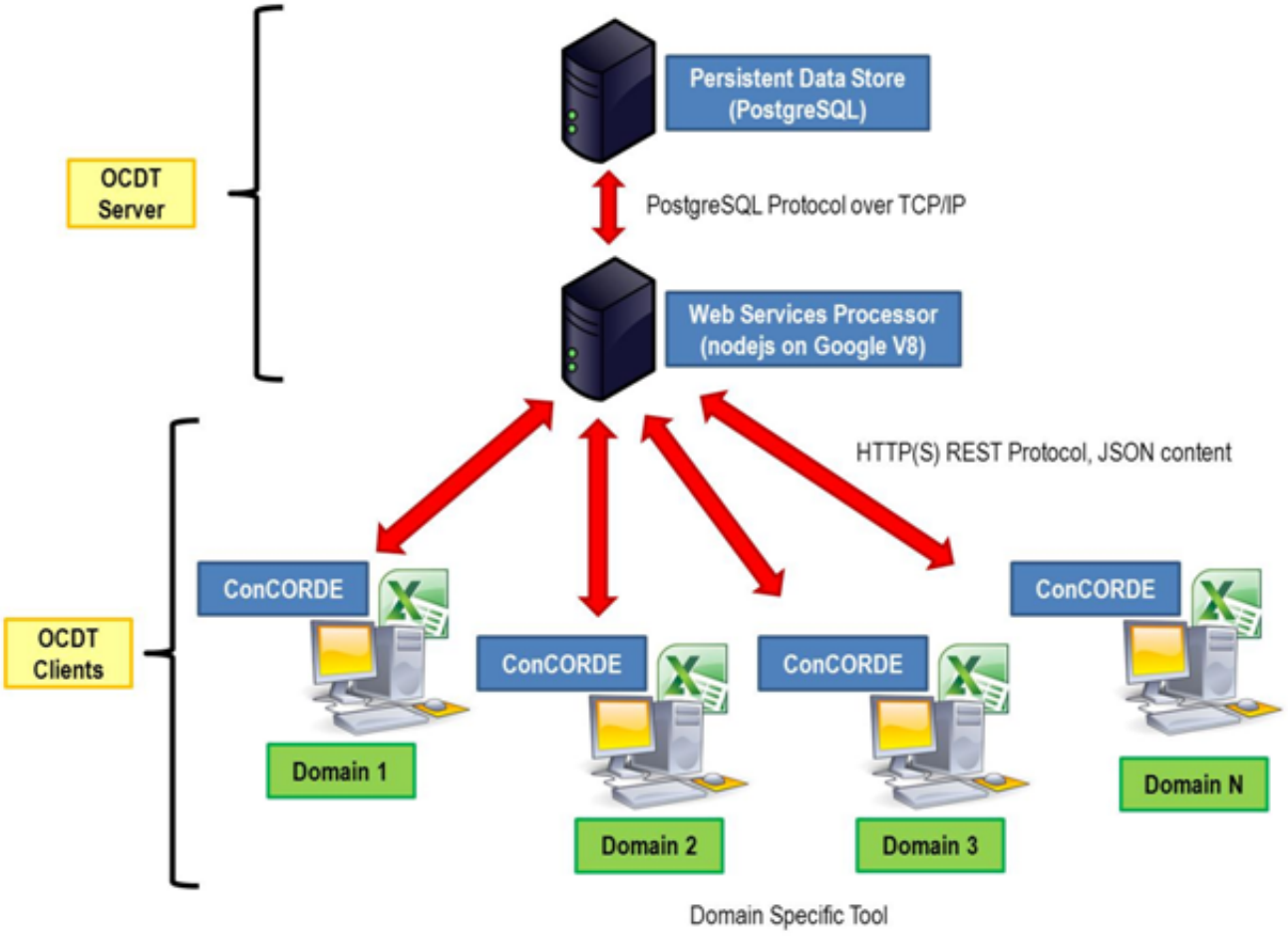

Figure 3.8: Open Concurrent Design Tool (OCDT) architecture overview.

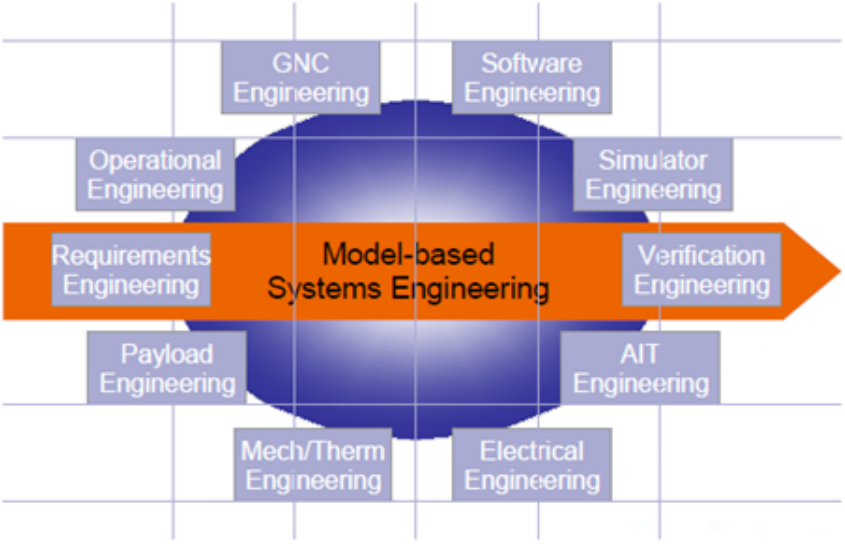

Figure 3.9: Engineering domains considered within the VSD project [21].

quired information. In this manner one worksheet per options is created and managed. Virtual Spacecraft Design (VSD) project represents another interesting research activity developed under the coordination of ESA [21]. This project has been conceived to define a model based methodology and a related environment, developed with the main purpose to improve the organization of engineering data at system level. The developed framework has been conceived to allow a more effective exchange of design parameters between different domains and their respective models through an object oriented approach [22]. This asset belongs to a larger project aimed to assess the application of MBSE philosophy within System Engineering, evaluating the benefits of a multi-disciplinary approach applied to the industrial phases of aerospace products [23]. The related activities have been thought from the fact that almost every project faces the problem of handling a wide quantity of data and information available at system and sub-system level and maintaining its coherence and consistency over all the development process. The effectiveness of System Engineering process is often limited by the presence of unstructured data and information not properly interlinked among all the disciplines involved in a project. The engineering domains considered within the proposed model-based methodology are briefly summarized in figure 3.9 [21].

Each domain often formalizes only some of the overall aspects of a system, focusing on the elements that mainly affect the related discipline. In this case the discipline models are defined for a specific simulation scenario and ensuring the consistency across such models is a challenging problem. The demonstration of the advantages achievable by such model based approach VSD project focused on three main key-points: engineering data representation, data exchange interfaces and process consideration. All the conceptual definitions and associations made within such project have been used to implement a software prototype to assess the actual effectiveness of the proposed environment. The overall software infrastructure is also called Virtual Spacecraft Engineering Environment (VSEE) and it is partitioned in three main building blocks: a reference database (Space System Reference Database - SSRDB), a design tool (Space Systems Design Editor - SSDE) and a visualization tool (Space Systems Visualization Tool - SSVT). An high level representation of such scheme is provided in figure 3.10. The functions provided by the developed framework are briefly represented in figure 3.11. In particular they are grouped with respect to the related capabilities associated with the Design Editor, the Visualization Tool and the Reference Database.

All the three building blocks provide different functionalities and capabilities that drive the project modeling activities, supporting the system engineering process. A more detailed description is however available from the project site [21].
VSD project basically involved the main European companies that work on space systems. They have joined such activities at different level and with various roles. The important aspect is that the conceptual data model that lays the foundation of the overall infrastructure has been defined through the contribution of the companies directly involved in the design and manufacturing of space products. In this way it was

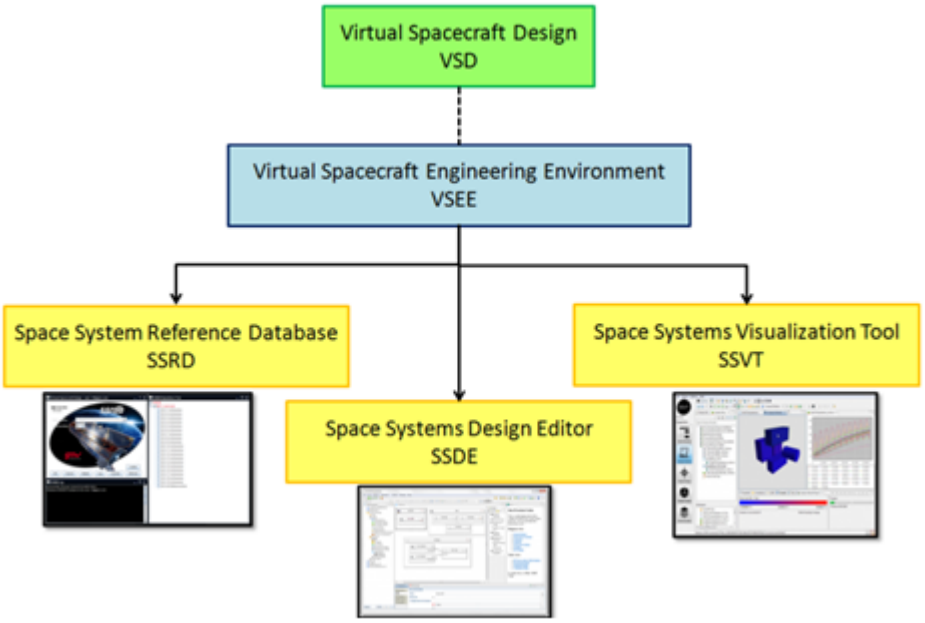

Figure 3.10: VSEE high level architecture.

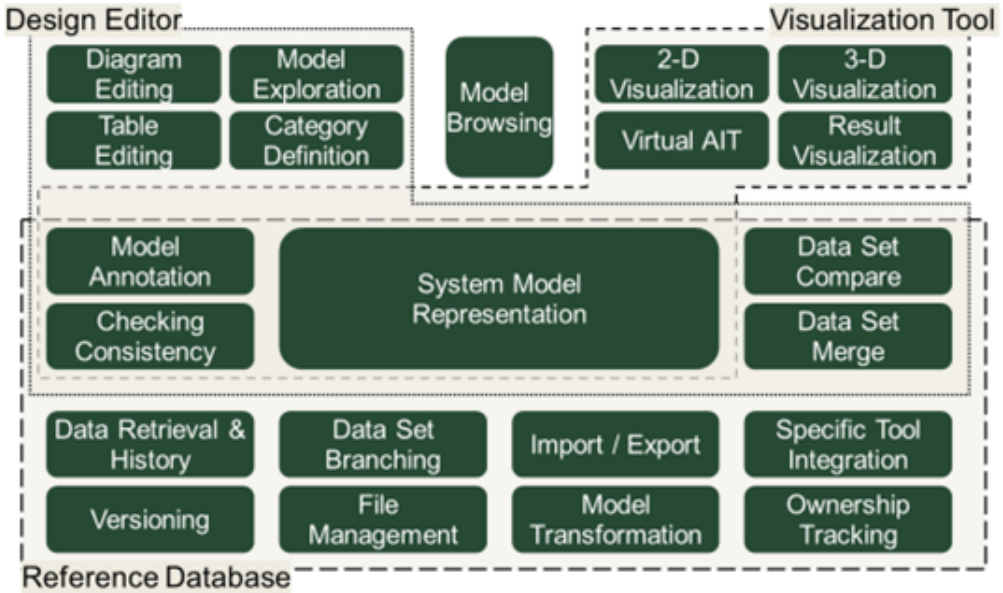

Figure 3.11: VSEE functions provided [21].

possible to start a confrontation on model-based methodologies, paving the way for future collaboration and definition of a common standard approach for the management of project data. Such topic draws the attention of the involved aerospace companies since the developed design and modeling methodologies will strictly affect future collaborations. It is not uncommon in fact that they work on the same space program due to the complexity of the related product and to the specific knowledge that each one possesses. A shared data model for the representation of system data and information will cover a key role in the next few years. For this reason both Astrium and Thales Alenia Space are currently interested in such topic and their attention has been highlighted by the contribution on Virtual Spacecraft Design project.

3.4.3 Centre National d'Etudes Spatiales – CNES

A concurrent design approach that follows the characteristics of the model based system engineering paradigm can also be identified in the research activities of CNES. In particular the developed architecture has been conceived mainly to design mission feasibility studies [24]. All the functionalities are provided through what can be defined as a Concurrent Design Facility (CIC - Centre d'Ingénierie Concourante). The focus of the related architecture and all the involved design process is represented by the assessment about the feasibility of space mission candidates, supporting the decision making activity. Such environment allows also investigating new technology concepts providing the basis for future developments and improvements. The architecture follows the standards defined within ECSS-ETM-10-25A, regarding the conceptual of engineering design models for data exchange. The system engineering information model has been conceived on the basis of the same standards (SEIM) which also provide useful guidelines for the definition of a reference data library (SERDL). All system information are managed through Excel worksheet description (equipments, sub-systems, payload, satellite, mission phases, etc.) while preliminary analysis and computations (as for example mass budget or power budget) are performed within the same Excel environment (for example through the proper execution of a macro/visual basic scripts). The various information and data are edited and monitored by different system level designers interacting with Excel worksheet. The same approach is used by the various disciplines (thermal, configuration, data-handling, AOCS, etc.) while concurrently the data are synchronized within such collaborative environment. Import/export functions are for example employed to proper manage the data flow between the central IDM-CIC workbook (Excel based repository) with domain specific tools. In this way CATIA files are managed through STEP files to extract the required information that are then stored within the workbook. Such example shows in particular the process through which the geometrical information are stored and managed. Visualization capabilities are also provided within such environment, allowing the plot of the data stored within the main central model. A structured XML file is build from the information available for the system model and such data are properly managed from a desktop application for example to help configuration analysis. This same environment is also used to highlight operational modes and power consumptions of the elements reported in the model. Concurrently to such application, the IDM-CIC workbook provides also some interesting utilities for a simple visualization through Google Sketch-Up tool.
Such a collaborative platform has been mainly conceived to support the early design phases providing technical instruments for the establishment of system budgets above all in the context of feasibility studies about mission design. The same approach is now under further developments and IDM-CIC roadmap is characterized by the investigation of a new architecture paving the way from the IDM-CIC version 2 to IDM-CIC version 3. The IDM-CIC V2 data exchange between subsystem and system models (both considering the definition of the related properties on an EXCEL workbook) is based on VBA API. The information collected within the single workbook is then interfaced with the STEP export and Google Sketch-Up through XML files. This approach is shared with all the disciplines involved within the project (data handling, thermal, AOCS, payload, configuration, system, etc…). IDM-CIC V3 is however based on the SERDL ECSS model with some adaptations with respect to geometrical descriptions with articulations, management of options and subsystem modes. This new approach foresees the definition of a shared folder location that contains the data exchange model as XML file. The single user computer accesses such data through API based on Microsoft .NET framework while an EXCEL GUI (also this based on Microsoft .NET framework) is used to manage and edit information, working on a related Workbook. The same EXCEL GUI is used to

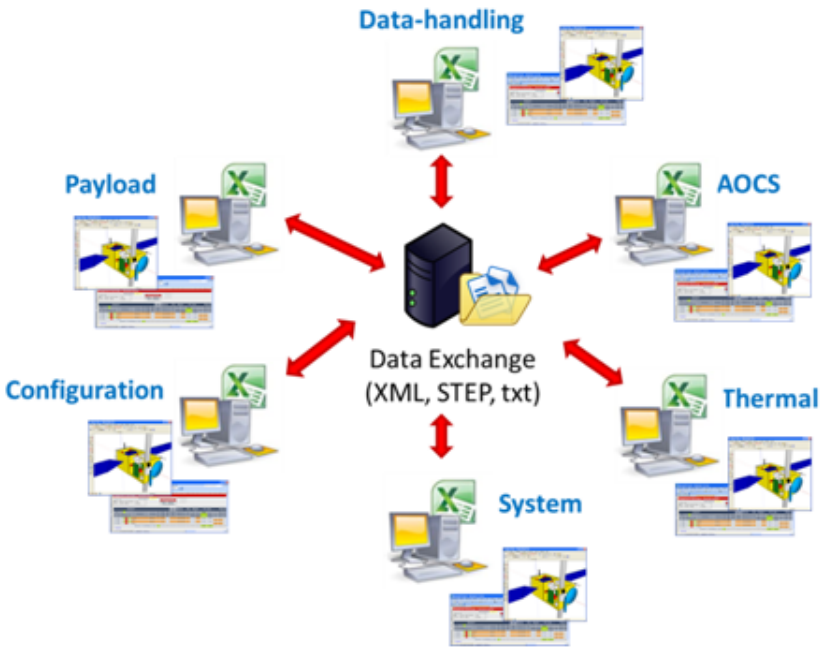

Figure 3.12: Simplified representation of CIC infrastructure (CNES).

export files through XML format as also manage the interfacing with external tools (STEP export, Google Sketch-Up, etc…). The current elements under developments regard tests, import/export units and STEP interface improving. Possible evolutions that are under investigation are represented by HTTP web services implementation (Web GUI and External tool) for the communication of the shared data repository (currently represented by XML file) and external environments. The user computer data exchange pattern is basically the same of IDM-CIC V2 while the main difference is represented by the definition of a server-client architecture for the management of the data contained within the XML file.
A simplified representation of the IDM-CIC architecture is reported in figure 3.12.

3.4.4 Thales Alenia Space

Thales Alenia Space (TAS) interest towards System Engineering is highlighted by the various research initiatives promoted in this direction. Main efforts are addressed to the definition of a concurrent engineering meta-data model and a formalization of multi-domain representation [25].
In this case a discrete amount of research efforts are addressed towards the investigation of MBSE methodologies and their integration within company processes. Such activities are also conceived to understand the possible enhancements that can be achieved with respect to collaborative environments. Different solutions are currently under evaluation to identify the better choice from a methodological viewpoint, considering also the possible infrastructural alternatives [26].
In particular TAS is introducing a tool for complex architectures modeling and functional representations (lacking in some of the tools currently used) called Melody Advance. Such tool has been developed on the basis of Thales Group company expertise about system architecture. Basically it is implemented as a variance of SysML language with the main purpose to improve its usage across system engineering. In the same way TAS is also introducing a tool for performances evaluation and verification called Arcadia, a Thales Group tool developed in Aerospace division.
Melody Advance tool is conceived for system and software architecture modeling and belongs more generally to System and Software development environment. Along with other tool Melody Advance is part of a more extended frame with the main idea to better control and support the design process from system perspective. For this reason a high level framework called Orchestra has been developed. Some of the principal objectives of such approach are reported in the following list:

- Automate routine activities

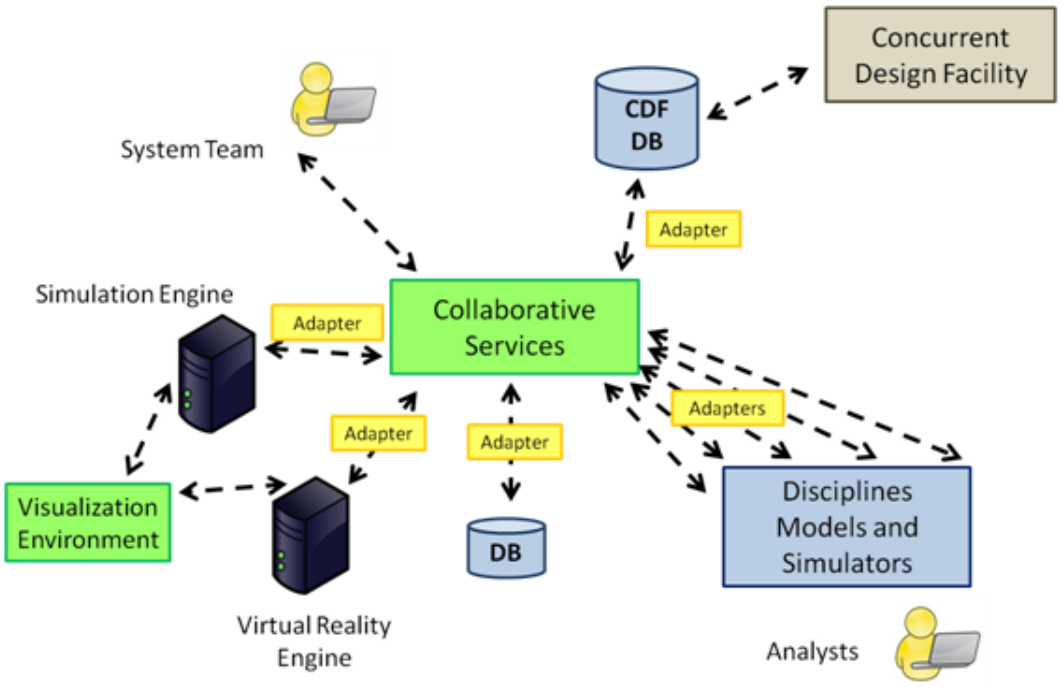

Figure 3.13: Conceptual overview of a collaborative environment infrastructure.

- Improve and ensure quality
- Interface legacy methods, products and services
- Provide and an effective set of services

In this context Melody Advance has the primary purpose to foster new system/software engineering methodologies, ensuring a support for Model Driven Engineering (MDE) approach (involving for example processes, activities, data and milestones). Currently Melody Advance is a user friendly tool that guarantees traceability and consistency between models and implements rules checking.
A conceptual representation of a possible infrastructure of a collaborative environment is reported in figure 3.13. In particular the features of such architecture are currently investigated within a research project led by Thales Alenia Space Italy. The name of the environment related to such project is DEVICE and more information about such initiative are provided in the following sections. A well-defined investigation of the capabilities that can be achieved with a model-based approach is represented by [107]. In this case the main advantages of such kind of infrastructures are evaluated, taking into account the potential integration between system data and complexity indices.

3.5 Benefits of MBSE

One of the main purposes animating the spreading of MBSE methodologies is represented basically by the capability to accelerate product design and manufacturing process. The same philosophy can also help the knowledge exchange among individuals and organizations but can and should improve the knowledge creation and externalization (as for example in the case of distributed cognition).
Another important benefit of MBSE methodology can be recognized in the document generation capability. This feature is directly related to the availability of system information within a central system model. This data can be directly queried to get the information needed and processed to generate documents and report without the effort that characterizes the traditional approach. Starting from the MBSE architecture it is possible to consider different strategies for the creation of the required document resources. In particular different system modeling tools are now providing various instruments and plug-ins for the support and definition of documents template. In this manner the information contained within the model

can be used to proper fill the structure defined within the template, obtaining potentially different report resources, from html to pdf formats. In this way engineers can spend less time in document generation activities, allowing the use of more efforts on modeling.
The document generation from system model information can be obtained through the implementation of proper defined scripts that process the data following the structure defined a priori in a certain template. Some evaluations of this capability have been done as for example in [27]. In this case the capability to generate documents has been analysed in the context of space mission application. In particular SysML modeling environment has been considered for this process and a conceptual architecture has been developed to build a document profile. The information is elaborated generating XML resources that are then further processed to obtain an html or pdf file. Internal generation scripts have been used to parse the SysML information to prepare the following creation of documents on the basis of defined template and style sheets. This approach has been used on different applications to evaluate the actual advantages. JPL Ops Revitalization project has used such generation capability as the JPL Integrated Model Centric Engineering for the document and reports creation (in this last case in the context of collaborative environment). Another application example is represented by the Mars Science Laboratory where this feature has been used to elaborate specific portions of operations processes and ground data system management.

3.6 Drawbacks and main needs of MBSE

A brief description of the main benefits that can be achieved through the application of MBSE methodology is presented in the section above but other advantages can also be found in the rest of the work. Despite all the positive considerations about model based approaches some drawbacks can arise from the integration and spreading of such methodology in the context of already consolidated processes and methods. In particular the use of such innovative approaches need to be tested on actual problems to avoid a worse result with respect to the traditional procedures. The integration of such methodology within engineering procedures established over the years requires for example the acceptance from those users that have matured a certain experience in a specific domain. From this viewpoint the training aspect must not underestimated to allow the integration within the current design methodologies. The overall infrastructure must also be properly defined to effectively manage an environment based on MBSE methods. One of the main problems is also represented by the efforts required in the definition of a well formalized infrastructure for modeling and data exchange. In particular the application of the concepts related to the MBSE philosophy often requires a wide conceptual work for the definition of the features and relationships of the objects that characterize system lifecycle. The use of a model based infrastructure requires the correct integration with system model information to provide useful capabilities to support the actors. Such aspect must be taken into account to properly manage user interactions with a model based platform, feature that basically does not affects the traditional modeling approaches and that may be the cause for some implementation issues that otherwise would not be occurred. The integration of MBSE methodologies within the design process can widely enhance the current development process but at the moment the application of such philosophy can not rely on a well defined set of tools, languages and platforms. Such aspects are fundamental for the actual integration of the related methodology with the current design processes. The availability of already defined instruments for the development of products strictly affects the spread of MBSE itself. Different solutions have been proposed by the organizations and industrial consortium to address this lack. SysML for example represents one of the possible solutions to face such kind of issues but other options can also be taken into account. In particular the main benefits and drawbacks of such language are considered in the following sections. A critical assessment of the main features about this approach and related application is provided in the following, proposing also alternative perspective for the development of strategies with respect to MBSE spreading. Also in this case the main benefits and drawbacks of the proposed approach and related technologies are highlighted, always keeping in mind that each possible solution has its own advantages and disadvantages. The final answer to all problems is in fact difficult to obtain with an unique tool but the identification of the limits of one approach with respect to another one can help to understand the direction to follow.

Chapter 4

Multidisciplinary Analysis

4.1 Introduction

As previously introduced this work proposes to investigate the integration of MDO methodologies within the context of a MBSE system model framework. In particular the issues that can arise from the considered approach are highlighted to better understand the possible improvements for the proposed infrastructure as well as also completely different solutions must be taken.
During the last few years the multidisciplinary integration between different engineering domains has started to be one of the most interesting and challenging research topics. Mathematical algorithms improvement and the concurrent implementation of object oriented software solutions seems to be increasingly well suited for the identification of optimal system configurations.
Multidisciplinary Design Optimization (MDO) is a methodology that includes all the activities related to the design of systems where strong influences characterize the interactions between disciplines. In these situations, now much more widespread in the development process than in the previous years, the designers are motivated to manage at the same time variables within several disciplines. For this reason MDO involves the coordination of different domain-specific analysis with the final aim to obtain more effective solutions, optimizing the configurations of complex systems.
The increasing complexity of space systems and the necessity to optimize the available resources have led to a deeper introduction of MDO methodologies within the product design and management process. In recent years these philosophy of design and implementation has gained increasing interest. The latter one is related above all to the possible advantages and future applications that this approach will allow to reach. Nowadays large amounts of systems, not only related to the aerospace area, are characterized by a close relationship of various disciplines. In this context the proper definition and setting up of the problem covers a key-role for a product successful realization. The suited management of the data involved, the correct analysis and good interpretation of simulation results contributes to the choice of the right direction. In particular we have often to deal with the definition of complex systems, facing in some cases the possible connections between various disciplines involved at different levels. It is in situations like this that the MDO provides useful instruments and methodologies to deal with complex design problems. Current engineering problems are increasingly characterized by a wide set of conflicting objectives that must be properly approached to avoid solutions that are less effective among the possible ones. Different methods can be used to employ multi-objective optimizations, and interesting applications can be found for the identification of concept alternatives, as reported in [114].
Systems projects currently involve an increasing number of design variables, constraints and objectives. Furthermore a group of design variables could be generally shared between different disciplines in this way. They are traditionally associated with consolidate dimensioning processes and such approach can help to improve the effectiveness of the overall process. For this reason their studies and analysis process become difficult to monitor, demanding a greater effort than the approach used traditionally in the past. The advantages of the considered methodology are mainly linked to the reduction of development time, allowing a more extensive evaluation of the design variables space. Closely related to this issue it is also possible to observe even a reduction of the costs of project activities. The automated process to properly

explore the design space with a series of well suited algorithms allows preventing the possibility to neglect certain system configurations. The latter ones could instead potentially represent the optimal solutions for the scenarios considered. An integrated design is then required to link up all the possible disciplines involved in system design, ensuring the access to the same data information and models.
The use of MDO is characterized by the following feature:

- Decomposition activities from the system model to multiple subsystems or discipline analysis.
- Development activities related to the generation of mathematical model and analysis. In this feature are included all the process that link "parent" system model with the "child" models and their interactions.
- Selection of the proper MDO formulation and algorithms on the base of the problem considered.
- Resolution of the MDO problem to finally generate the solutions on the base of the set-up considered.

The main distinction about the feasibility of the solutions explored depends on the number of disciplines considered. One possibility is to manage multiple disciplines concurrently, trying to move towards a better design and re-establishing feasibility. The other approach is to consider instead individual discipline feasibility. In the previous MDO approaches the collaborative optimization techniques define the decomposition of system into smaller units that can be individually optimized and then linked to the system. The system level optimizer sets the design objective, generating the interdisciplinary compatibility constraints that are then submitted to the various subspace optimizers. These ones are generally grouped according to the different domain-specific disciplines. Therefore the single subspace optimizer must satisfy the assigned objectives ensuring the interdisciplinary compatibility, considering the analysis results that are generated. This approach reflects one of the initial framework that employs MDO methodology and an example can be found in [32] where it was analyzed an underwater exploratory vehicle.
Approaches similar to the one presented can be often found in the literature, with only slight differences between each other. The main management process is basically the same. Namely there is a set of design alternatives that require to be analyzed and the same variable potentially can affect several domain-specific models and tools. In this situation often different iterative cycles are required to reach the convergence of the specific-field models for a particular configuration of the variables set. Once the models physical meaning is ensured (through the required convergence and assuming the correctness of the mathematical formulation considered) is then extracted the results that allow to generate the indexes and the quantities for the evaluation of system performances. In literature different researches are addressed towards the investigation of a wide set of topics that are strictly related to the improvement of the actual multidisciplinary analysis techniques applied to complex systems (interesting examples can be found in [115], [116], [117] and [118]).
One of the active research fields related to the MDO and considered in the previous study concerns the creation of surrogate models. Reduction of the computational time represents one of the interesting features for the future applications.

4.1.1 Current needs of MDO techniques

Solving techniques are often chosen on the basis of the specific needs and the available resources. Models with different fidelity levels are used to face engineering problems in different manners. Low fidelity models can be represented for example by aerodynamic panel codes or equivalent-plate structures codes while medium fidelity ones can be identified with Euler CFD models, FEM structural models or axisymmetric propulsion codes. High fidelity models concern instead Navier-Stokes CFD codes, adaptive FEM models or 3D propulsion models for example. They are used differently depending in particular on the level of details required by the current design phase and their integration within the same collaborative environment represents a challenging research topic. In the past the management of multidisciplinary design analyses has been done through system-level coordination, partitioning the original problem into different sub-problems (not necessarily disjoint). Each sub-problem had its constraints and objective function. A

collaborative optimization approach ensures a parallel and autonomous processing of the disciplines. It allows also managing the various elements more consistently in the design environment and organizational structures. The drawbacks of such methodologies are related to the poor robustness and convergence characteristics about the evaluation of overall system performances.
A deeper integration between systems analysis and MDAO (Multidisciplinary Design Analysis and Optimization) processes/methods covers a key role with respect to the achievement of an effective product ([109]). Such need is widely underlined by various research initiatives available from the literature, as can be found in [112] and [113]. In the last years some efforts have been addressed towards a clearer representation of MDAO infrastructures with the final objective to achieve a formal representation ([132], [133]).

4.1.2 MDO architectures

The integration of MDO methods requires a well understanding of the architecture that must be selected for the implementation of analysis process. The term MDO architecture identifies how the simulation blocks, analysis elements and overall process flows are related between each other. Such definition refers both to problem formulation and the organizational and algorithmic strategy to solve the problem. Different MDO architectures are available from the literature but each one is often presented within a specific research context while a common and shared standard for the description of such patterns could be very useful. An interesting survey is provided in [33] where clear definitions and a standard representation are proposed to basically describe MDO architectures. In particular the considered work presents a unified description about MDO architectures, providing a set of mathematical concepts and notations both for problem formulation and solution strategy. This approach has been introduced since the same notation will be considered for the presentation of the most common MDO architectures in the following section. A brief description of the main characteristics of such a unified representation is introduced to better understand the following diagrams. Other studies have faced the problem of providing a unified description of MDO architectures such as [34] where a linguistic approach called Reconfigurable Multidisciplinary Synthesis (REMS) has been proposed. This work provides useful guidelines above all for implementation within the computational environment but does not employ visual references on how the input problems are managed.
Often the choice of the MDO architecture depends on many parameters such the problem characteristics, design environment and available software tools. These elements strongly influence the problem formulation and the solution strategy employed which on the other side affect the resources employed to identify a design solution. Two important concepts must be clearly understood before the visual representation of diagrams is reported. The first one is represented by the data flow among the various problem components while the second one is the sequence of operations that must be accomplished to find the design solution. Often the description of both these different concepts is done through the use of same block diagrams, flowcharts and algorithms representation, reducing the capability to clearly understand of the considered MDO methodology.
The considered framework includes a common mathematical notation for the formulation of problems and diagrams that describe the solving process. The same approach will be used in the current work when needed. Some useful definitions are introduced in the following lines with the main purpose to well clarify the terms and concepts that will be used in the current work.
A design variable represents a variable that in the context of MDO problem is always under the control of the designer. In particular the design variables can belong to a specific discipline or to multiple ones that share some common features. This aspect is taken into account in the definition of the design variables vectors since different arrays are defined for the single discipline with its specific variables. A common design variable vector is instead used for the shared variables among the various disciplines. The vector that contains the design variables belonging to discipline **i** is represented by x_i while the vector that stores all the design variables share with at least two disciplines is denoted as x_0.
A Discipline Analysis (DA) identifies a simulation addressed to study a particular aspect of a multidisciplinary system. The execution of a discipline analysis involves the solving of a system of equations (often identified as the disciplinary equations) which are used to compute a set of disciplines responses. The last ones

Table 4.1: Mathematical notation for MDO problems.

Symbol	Definition
$\boldsymbol{x}$	Vector of design variables
$\boldsymbol{y}$	Vector of coupling variables (outputs from DA)
$\bar{\boldsymbol{y}}$	Vector of state variables (variables used inside only one DA)
$\boldsymbol{f}$	Vector of objective functions
$\boldsymbol{c}$	Vector of design constraints
$\boldsymbol{c}^c$	Vector of consistency constraints (between target and state variables)
$\boldsymbol{R}$	Governing equations of a DA in residual form (DA constraints)
N	Number of disciplines
$\boldsymbol{n}_{()}$	Length of given variable vector
$\boldsymbol{m}_{()}$	Length of given constraint vector
$()_0$	Functions or variables that are shared by more than one disciplines
$()_i$	Functions or variables that apply only to discipline i
$()^*$	Functions or variables at their optimal value
$\widetilde{()}$	Approximation of a given function or vector of functions
$\widehat{()}$	Independent copies of variables distributed to other disciplines

are often called state variables and their management can be driven or not by the optimization process, depending on the problem formulation. State variables referring to the discipline *i* are contained within the vector y_i. Basically some state variables that are computed in a single discipline are also required by other discipline/disciplines in a multidisciplinary system. Such variables can be defined as coupling variables and they are represented in the same notation of state ones. The computational process often is characterized by the possibility to run some codes in parallel with other simulations. In this case copies of the vector containing the design and state variables are made to allow DA independent execution. These copies are identified with the superscript $\wedge$ in this work since such elements are often called coupling targets. The target state variables vector of the discipline **i** is then represented by the notation$\widehat{y_l}$. This object is used to share the state variables provided by the discipline i among the disciplines that need at least one of the state variables contained within the vector y_i. The consistency between the state variable y_i and the related target one $\widehat{y_l}$ must be ensured through the definition of proper constraints that are added to the problem formulation. The mathematical notation used in the current work is reported in table 4.1 (all definitions are exposed with more details in [35]).

MDO problems can be defined following the pattern of a particular formulation and on the basis of the chosen one the related constraints, equations and relationships are formalized. All MDO architectures solve optimization problems that can be derived from what is known as the All-at-Once (AAO) formulation which includes all the analysis equations, design objectives, design constraint and consistency between the inputs and outputs coming from the various DA. Such formulation can be expressed as in the following lines.
All-at-Once formulation:

$$
\begin{aligned}
&\text{minimize:} && f_0(x,y) + \sum_{i=1}^{N} f_i(x_0, x_i, y_i) && \\
&\text{with respect to:} && x, \hat{y}, y, \bar{y} && \\
&\text{subject to:} && c_0(x,y) \geq 0 && \\
& && c_i(x_0, x_i, y_i) \geq 0 && \text{for } i = 1, \cdots, N \\
& && c_i^c = \hat{y}_i - y_i = 0 && \text{for } i = 1, \cdots, N \\
& && R_i(x_0, x_i, \hat{y}_{j \neq i}, \bar{y}_i, y_i) = 0 && \text{for } i = 1, \cdots, N
\end{aligned}
$$

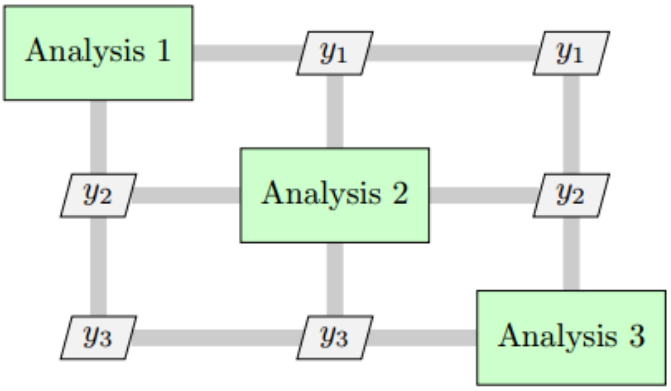

Figure 4.1: N^2 chart example [33].

Where *N* represents the total amount of the involved disciplines and the vectors x,y, $\widehat{y}$ and $\overline{y}$ are defined as:

- $x = [x_0^T, x_1^T, \cdots, x_N^T]^T$
- $y = [y_1^T, \cdots, y_N^T]^T$
- $\hat{y} = [(\hat{y}_1)^T, \cdots, (\hat{y}_N)^T]^T$
- $\hat{y} = [(\bar{y}_1)^T, \cdots, (\bar{y}_N)^T]^T$

The development of MDO strategies often starts from a clear understanding about the relationships that characterize the information exchange between all the various elements defining a multidisciplinary environment. In particular visual representations like N^2 charts allow clarifying the connections among all the considered objects. An example of an N^2 is proposed in figure 4.1 where three coupled analysis are conceptually considered.

This diagram has been conceived in the context of System Engineering to clearly identify in a better way the dependencies of all the components that characterize our problem. In particular such diagram can be used both for the analysis of the relationships between topological/physical components and activities/procedures. In particular the main aim of such representation is the identification of the inputs/outputs needed/provided by the single element in the context of the overall architecture. Similar diagrams are also represented by the Design Structure Matrix (DSM) which is a network modeling tool used to represent the elements comprising a system and their interactions, thereby highlighting the system architecture ([36]). In particular DSM diagrams can be classified in two ways: as static DSM, which basically are equivalent to N2 charts, and time-based DSM. In the first case the representation does not contains information related to the time dependencies of the involved elements while in the second case the objects are disposed along the matrix structure taking into account for the temporal relationships that realize through the design process. In time-based DSM the objects that are involved in the early phases of a certain process are placed towards the upper left corner of the matrix while the following ones are located in the lower right corner as time proceeds. Examples of DSM matrices are reported in figure 4.2.

In the specific context of MDO problems the components represented in DSM diagrams can be disciplines analyses, objective and constraint functions, optimizers, surrogate models or other computational elements. The interactions between such elements are mainly represented by the exchange of data such as design variables, function values or state variables. The matrix visualization is often characterized by the fact that the interactions of a specific element with itself are meaningless and the diagonal cells of the table are not characterized by any information since they not show any dependencies. In other cases the same diagonal sub-matrix positions can be occupied by specific elements directly enhancing the relationships with the other objects in the matrix through rows and columns information flows. The most widespread convention considers the inputs for a certain element (represented on the diagonal) placed on

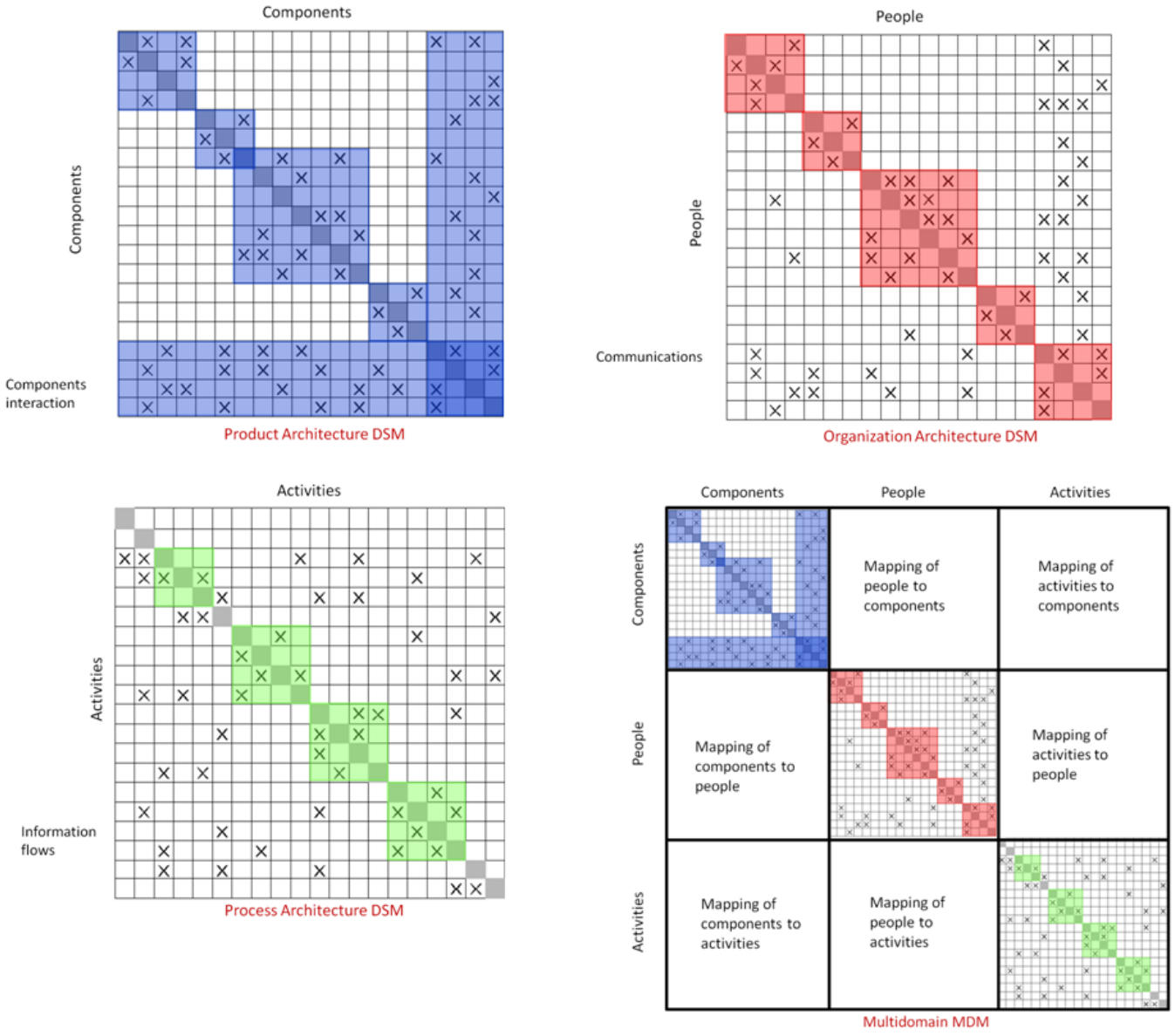

Figure 4.2: Examples of DSM concerning Product Architecture, Organization Architecture, Process Architecture and Multidomain matrix [36].

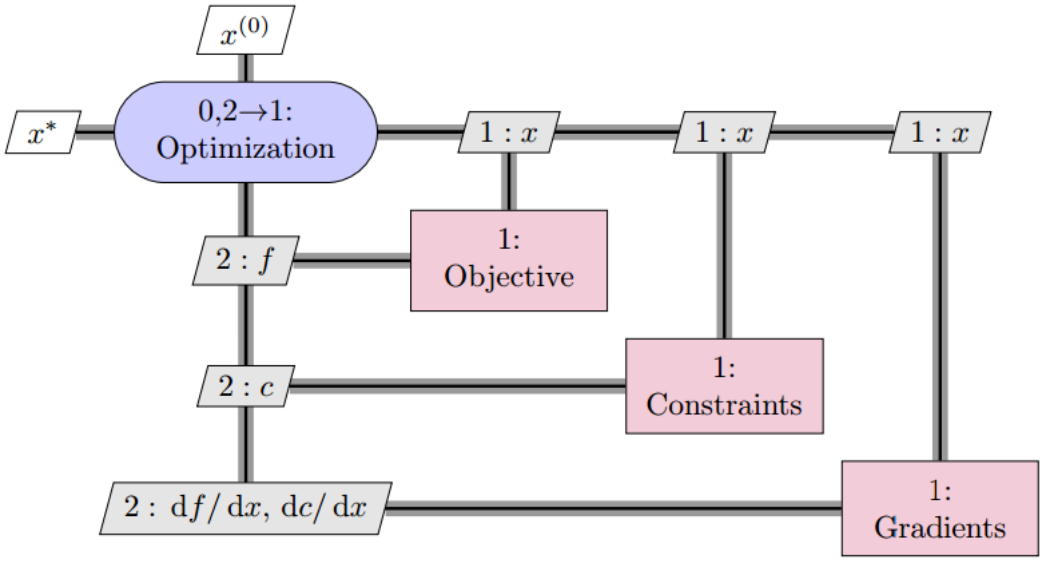

Figure 4.3: Simple example of gradient-based optimization process [35].

the same column (data flowing towards the object itself) while the outputs are provided on the row. Terms like feedback or feed-forward become meaningful when the interfaces are considered. Generally the interactions that are highlighted in the lower triangular part of the matrix are identified as feedbacks while those in the upper triangular region represent the feed-forward relationships. Such classification strictly depends on the flow convention for inputs and outputs while the main concept to clearly understand is how a certain element interacts with other objects, enhancing the data required and the information provided. The same system (or MDO problem) can be represented with different DSM only changing the order and disposition of the various element on head row and column, maintaining however the dependencies between the element themselves. Changing such disposition can be done to better manage the overall process, reducing for example the time required to solve an MDO problem. In this case sequencing and clustering algorithms are employed to obtain such result. Some research initiatives are currently evaluating the extension of DSM for the description of multidisciplinary design, analysis and optimization processes (an interesting study is available in [82]).
The diagram notation uses gray connection lines to show the dependencies related to the data exchanged between the various elements. Such lines allow to understand which inputs and outputs are involved but do not provide information about the order in which of object operations or analyses are executed. This last aspect is managed through the introduction of a system of additional lines identified with black, thin connections while the execution order is highlighted with a numbering schema. In this way it is possible to understand the process order following the sequence steps while potential computational loops are denoted with other indexes nesting from the root one. When certain components can be executed in parallel then the same number is used as entry for both the components. The main execution scheme can be interfaced with external data as for example an initial design vector (starting point) in the case of an optimization problem while at the same time the overall process produces output information like the optimal solution identified. The same notation can also be used to clearly represent the process flows related to an optimization strategy. A gradient-based optimization procedure can in fact represented as in diagram 4.3.

One of the most challenging processes in the context of system engineering and multidisciplinary analysis is represented by the determination of system full state. In particular such process is represented by the identification of a complete system state that is consistent with all the disciplines and analysis involved. This analysis becomes particularly complex when different variables are shared among various disciplines. In this case the consistence between models coming from different disciplines and often also from different simulation environments is not so easy to manage. An example of multidisciplinary analysis is represented by the Gauss-Seidel MDA, where the main aim is basically addressed to the evaluation of overall system state, trying to reach a whole consistent condition. Such analysis in the case of three disciplines is reported

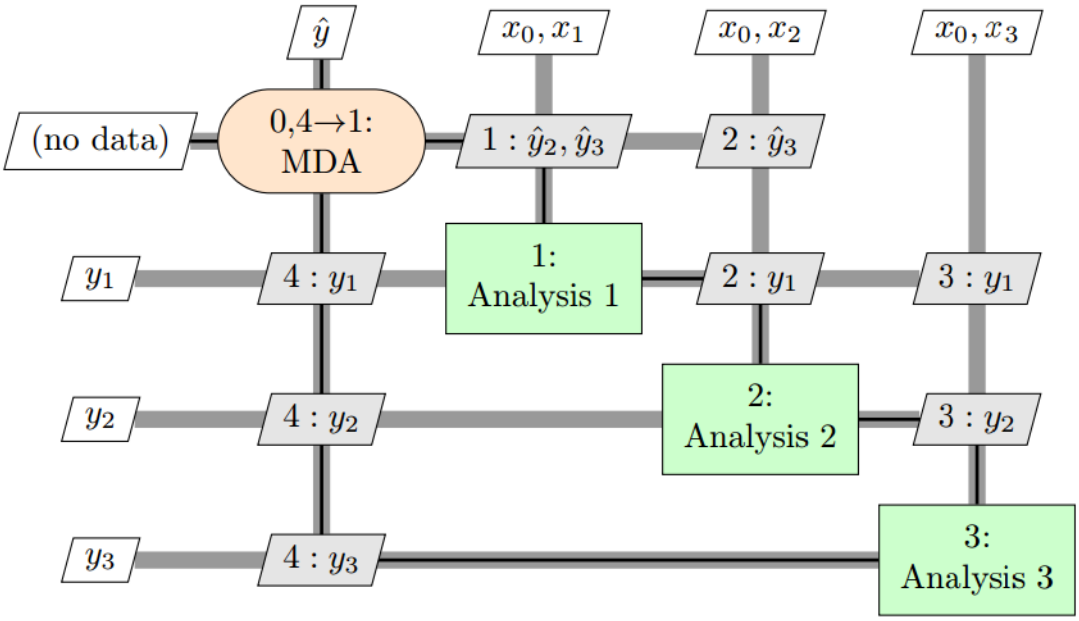

Figure 4.4: Gauss-Seidel MDA architecture for three coupled analyses [35].

in figure 4.4.

The same notation will be used in the following lines to describe some of the most famous MDO architectures but the same approach can basically be adopted to represent other multidisciplinary patterns. The description of the following architectures will be considered as preparatory to better understand the integration of multidisciplinary design framework with the modeling environments. The MDO architectures that we consider are reported in the following list but it is possible to find also other typologies in the current literature.

- Multidisciplinary Feasible (MDF) architecture
- Individual Discipline Feasible (IDF) architecture
- All At Once (AAO) architecture
- Collaborative Optimization (CO) architecture
- Bilevel Integrated System Synthesis (BLISS) architecture – in particular the BLISS-200 variant

The MDF architecture deals with the interaction between different disciplines all coupled together where an MDA analysis (which characteristics are briefly introduced in the previously lines) is performed to evaluate a consistent overall system state for a certain set of design variables. Each design set must be properly evaluated through MDA since the optimization algorithm does not know a priori the feasibility of such choice. In particular each design set must converge to a feasible state, if possible, before the optimization techniques proceeds with the following iterations/function evaluations. The problem formulation related to such architecture is expressed with the following relationships:

$$\begin{aligned}
&\text{minimize:} && f_0(x, y(x)) + \sum_{i=1}^{N} f_i(x_0, x_i, y_i(x_0, x_i, y_{j\neq i})) \\
&\text{with respect to:} && x \\
&\text{subject to:} && c_0(x, y(x)) \geq 0 \\
& && c_i(x_0, x_i, y_i(x_0, x_i, y_{j\neq i})) \geq 0 \qquad \text{for } i = 1, \cdots, N
\end{aligned}$$

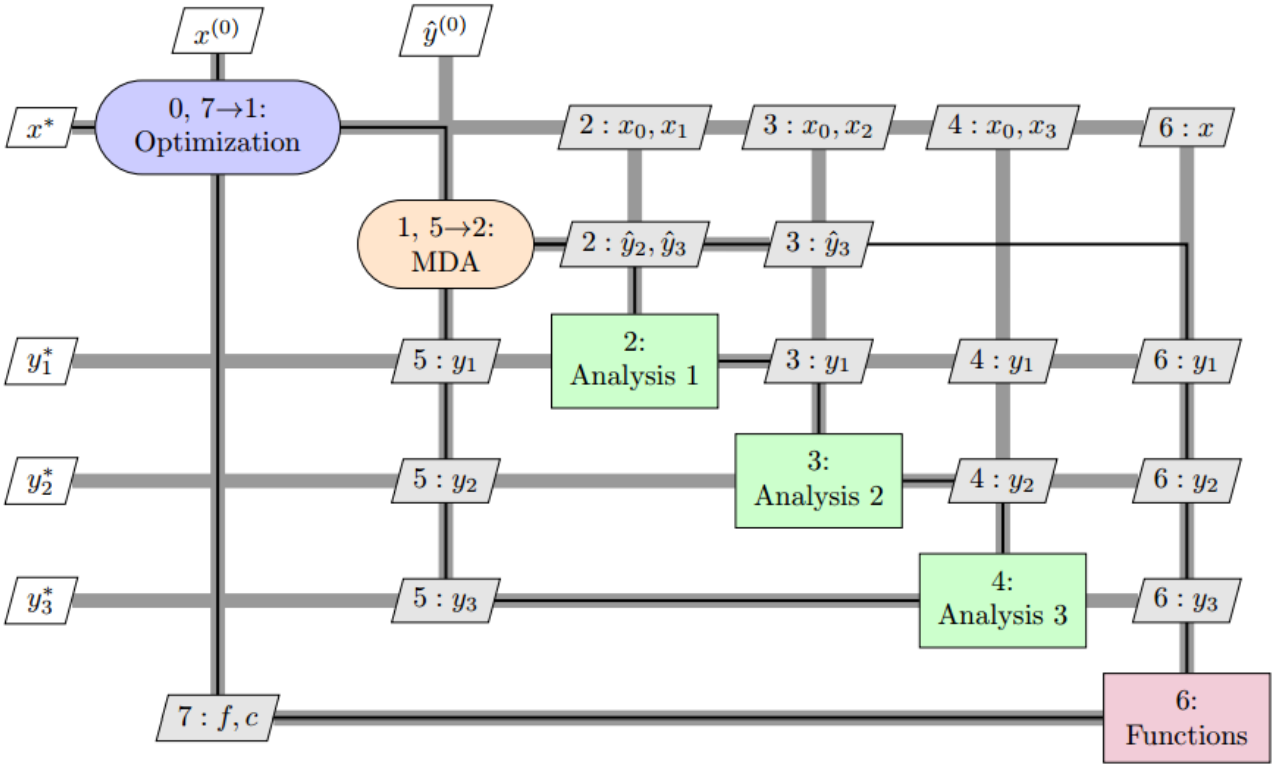

Figure 4.5: MDF architecture with Gauss-Seidel MDA integration for three coupled analyses [35].

The analysis and consistency constraints involving all coupling variables have been removed from the formulation since they are managed in the nested MDA block. In this manner they are automatically satisfied for each iteration while the optimizer deals only with the choice of the design variables set. It is important to underline the fact that the same architecture can be implemented using different solution strategies since the diagram representation considered has generic visualization purposes. The main structure of MDF process is reported in figure 4.5, where three analysis blocks have been considered in the example.

Another interesting scheme is represented by the IDF architecture which problem formulation can be resumed in the following pattern.

$$
\begin{aligned}
&\text{minimize:} && f_0(x, y(x, \hat{y})) \\
&\text{with respect to:} && x, \hat{y} \\
&\text{subject to:} && c_0(x, y(x, \hat{y})) \geq 0 \\
& && c_i(x_0, x_i, y_i(x_0, x_i, \hat{y}_{j\neq i})) \geq 0 && \text{for } i = 1, \cdots, N \\
& && c_i^c = \hat{y}_i - y_i(x_0, x_i, \hat{y}_{j\neq i}) = 0 && \text{for } i = 1, \cdots, N
\end{aligned}
$$

The elimination of disciplines analysis constraints $R_i(x_0, x_i, y_{j\neq i}, \overline{y_i}, y_i) = 0$ is allowed thanks to the use of implicit function theorem since the $\overline{y_i}$ and y_i are not managed independently but are now bounded to each other. They in fact become functions of design variables and coupling variable copies. The same definition of such architecture can also be identified with distributed analysis optimization and optimizer-based decomposition but all have the same theoretical problem formulation. Each iteration is characterized by the exact resolution of disciplines analysis equations and this condition all the coupling variables are now implicit functions of design variables and coupling variable copies. In particular in this case the individual disciplines are not coupled together when the system has been analyzed. Coupling variable copies are however used to share information among disciplines while consistency constraints are checked to control the correctness through the Coupling variables across the disciplines domains. Basically the IDF architecture is characterized by the fact that the individual DA resolves the analysis constraints directly on their own. In

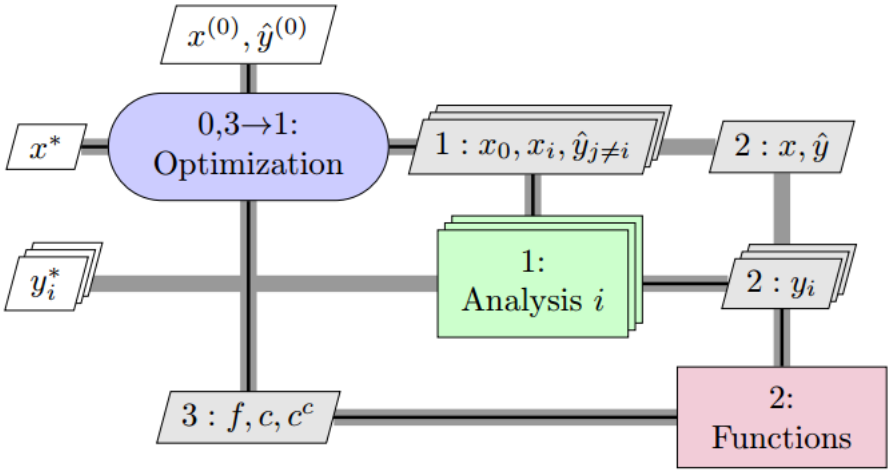

Figure 4.6: IDF architecture [35].

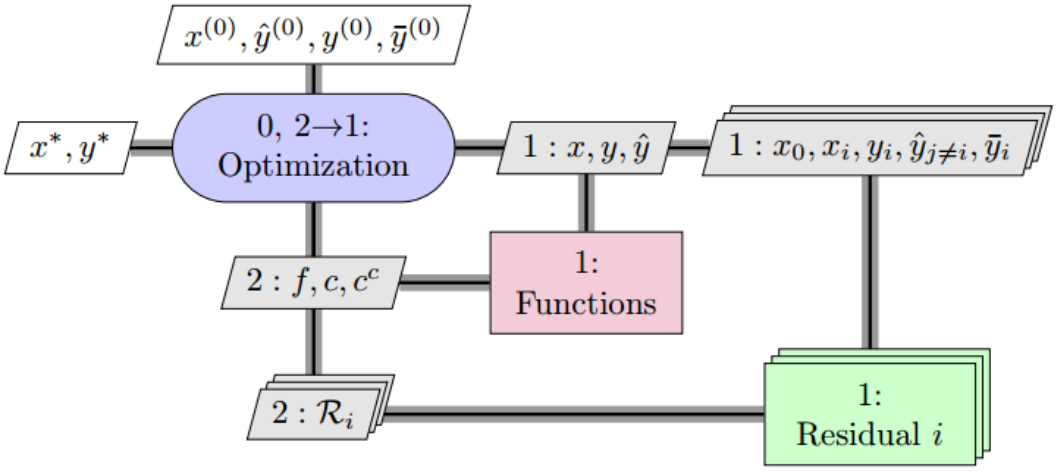

Figure 4.7: AAO architecture [35].

this way all the DA can be evaluated in parallel before the control will be passed again to the main algorithm for the next high level iteration. Besides the actual parallel execution the DA is strictly dependent on the hardware implementation for the overall cycle. The concept of parallel execution considered stands more generally for the fact that all the DA involved are not necessarily dependent on each other. They can be run parallel but also sequentially (on the basis of the available resources) but the main important things to understand is the fact that they can be executed each iteration without particular needs from the other elements (the consistency with the coupling variable copies is resolved internally).
A conceptual example of the IDF architecture is reported in figure 4.6.

Another well-known architecture is represented by the All-At-Once (AAO) scheme. The related formulation has already been considered in the first part if this brief introduction. The corresponding diagram can be seen in figure 4.7.

In this case the residual equations are not managed implicitly as in IDF architecture where such equations are used to resolve internally the dependence between state variables and copies of the coupling variables. In AAO architecture the residuals of the governing equations are in fact managed as constraints in the related problem formulation. The evaluation of residuals can be done in parallel or in sequential manner (on the basis of the available resources) since each one is independent with respect to the other. In AAO pattern the computations of objective and constraint function can be done concurrently with the residuals evaluations since all the required data are available from the algorithm main driver at each iteration. In the case of IDF architecture the evaluation of objective and constraint functions is done instead after the computations of the various disciplines analyses. Such a situation limits the possibility to run

these processes concurrently with the functions evaluations.
The Collaborative Optimization (CO) architecture is one of the other interesting pattern considered in this section. In particular such scheme shows well-different characteristics with respect to the previous formulations. Its main purpose is represented by the management of each discipline with a greater autonomy, using decomposition and coordination patterns for such objective. Different sub-problems are defined for each discipline in addition to the main system multidisciplinary problem. In literature two formulations about CO architectures can be found and the second one is considered since it is the most frequently used. The related formulation is expressed in the following lines:

$$\begin{aligned}
\text{minimize:} \quad & f_0(x_0, \hat{x}_1, \cdots, \hat{x}_N, \hat{y}) \\
\text{with respect to:} \quad & x_0, \hat{x}_1, \cdots, \hat{x}_N, \hat{y} \\
\text{subject to:} \quad & c_0(x_0, \hat{x}_1, \cdots, \hat{x}_N, \hat{y}) \geq 0 \\
& J_i^* = \|\hat{x}_{0i} - x_0\|_2^2 + \|\hat{x}_i - x_i\|_2^2 + \|\hat{y}_i - y_i(\hat{x}_{0i}, x_i, \hat{y}_{j\neq i})\|_2^2 = 0 \qquad \text{for } i = 1, \cdots, N
\end{aligned}$$

Once the overall system formulation has been define the discipline sub-problems are defined with the following relationships (there is one sub-problem for the i considered discipline/analysis).

$$\begin{aligned}
\text{minimize:} \quad & J_i(\hat{x}_{0i}, x_i, y_i(\hat{x}_{0i}, x_i, \hat{y}_{j\neq i})) \\
\text{with respect to:} \quad & \hat{x}_{0i}, x_i \\
\text{subject to:} \quad & c_i(\hat{x}_{0i}, x_i, y_i(\hat{x}_{0i}, x_i, \hat{y}_{j\neq i})) \geq 0
\end{aligned}$$

The J_i functions are introduced to ensure the consistency between the variables copies. Each discipline sub-problem is addressed to the minimization of data inconsistency and any local objective. At the same time the overall system algorithm deals with the minimization of system objective and the consistency among the various singularly optimized discipline sub-problems. The main drawback related to such an approach is represented by the fact that each domain sub-problem must be solved once to complete a single iteration of the multidisciplinary system problem. This architecture can also be identified as a bi-level optimization approach. A conceptual representation of CO architecture is reported in figure 4.8.

The last example of MDO schemes considered is represented by the Bilevel Integrated System Synthesis (BLISS-2000 variant) architecture. Such example is quite similar to the CO typology since the overall optimization problem is decomposed into system and single discipline sub-problems. The main difference with respect to the CO architecture is related to the fact that the disciplinary sub-problems are managed through the use of surrogate models. In particular such models are used to analyze the influence of the coupling design variables on the optimality of the single sub-problem. The related formulation is:

$$\begin{aligned}
\text{minimize:} \quad & f_0(x, \tilde{y}(x, \hat{y})) \\
\text{with respect to:} \quad & x_0, \hat{y}, w \\
\text{subject to:} \quad & c_0(x_0, \tilde{y}(x, \hat{y}, w) \geq 0 \\
& \hat{y}_i - \tilde{y}_i(x_0, x_i, \hat{y}_{j\neq i}, w_i) \qquad \text{for } i = 1, \cdots, N
\end{aligned}$$

The sub-problems formulations follow instead the following relationships

$$\begin{aligned}
\text{minimize:} \quad & w_i^T y_i \\
\text{with respect to:} \quad & x_i \\
\text{subject to:} \quad & c_i(x_0, x_i, y_i(x_0, x_i, \hat{y}_{j\neq i})) \geq 0
\end{aligned}$$

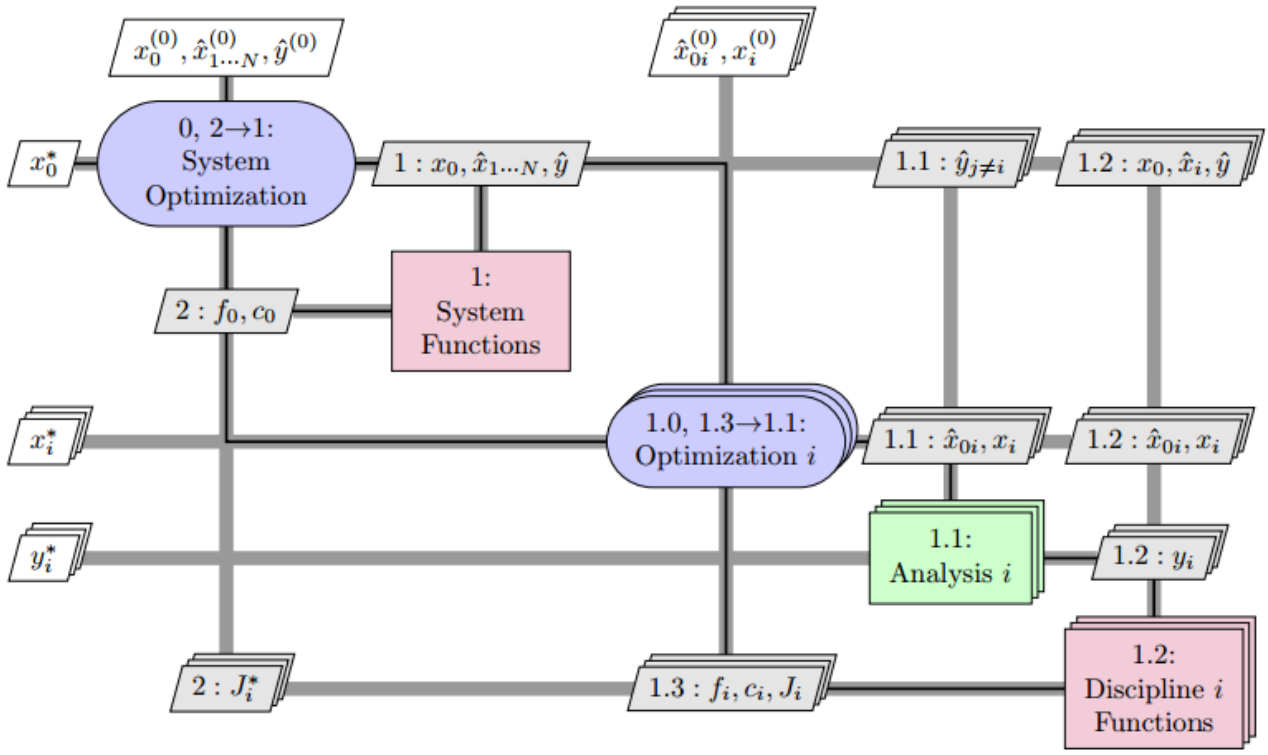

Figure 4.8: CO architecture [35].

Where w is the vector of weighting coefficients and it is chosen on the basis of the control preferences over the state variables. These coefficients are in fact directly related to the global objective and they affect how quickly the sub-problems optimal solution is found. The diagram of BLISS-2000 architecture can be seen in figure 4.9.

One of the characteristic aspect of this architecture is represented by the fact that the various DA return the data not directly to the system sub-problems since the information are first used to update surrogate models.

The proposed unified notation is a well-defined instrument for a clear and formal representation of MDO architectures as can be seen from the proposed example. The same terminology will be used in the current work for the description of multidisciplinary problem setting and architecture.

4.2 Available tools for MDO problems

The management of multidisciplinary design problems has attracted the interest of a quite wide range of software houses and commercial initiatives. In particular the current market provides different solutions for the commercial Process Integration and Design Optimization (PIDO) software [30]. They offer the capability to interface different kind of external solvers which are often based on heterogeneous analysis environments. Optimization capabilities can be also provided within the individual analysis environment but such embedded utility are often not properly conceived to handle complex multidisciplinary problems. Different simulation environments are however addressing some of their efforts in this direction with the final aim to provide robust functionalities in a unique framework.

Optimization toolkits that are embedded within the same modeling framework can be found within some analysis suites. A short list of the available embedded tools for optimization analyses is reported in the following only to provide actual references to the current approaches.

Optimization functionalities are available within SolidWorks (Design optimization study), Matlab/Simulink (Optimization Toolkit), Altair Hyperworks (Optistruct, Hyperstudy), etc.

More interesting results can however be obtained through the integration of multiple external solvers. In particular the connection with external analysis environments becomes fundamental when complex

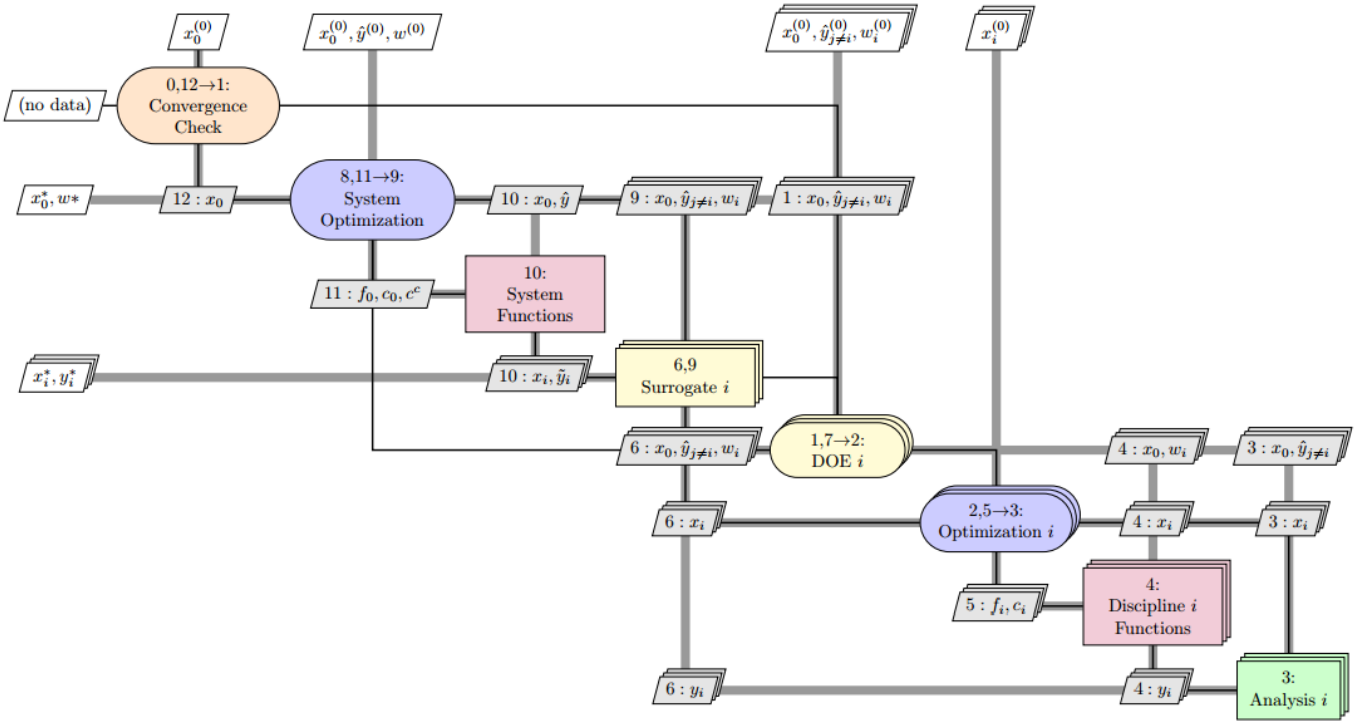

Figure 4.9: BLISS-2000 architecture [35].

multidisciplinary design problems are faced. Some domain-specific analyses are often managed through validate frameworks but their capabilities do not cover all the possible simulation scenarios. Commercial tools properly behave when the specific analysis domain is approached but complex scenarios often need to consider multiple physical aspect for the system under development. For this reason the tools used for particular simulations become not suitable for other ones. Within this context the use of a PIDO software can help to model and analyze more complex situations, allowing the interaction between proprietary tools that often are not explicitly conceived to interface with other software. Some examples are reported in the next lines.

The **CONSOL-OPTCAD™** tandem is for example a tool for interactive optimization-based design of a large class of systems and has been developed from the Institute for System Research (University of Maryland). The essential requirements are that a simulator be available for evaluating the performance of instances of the system under consideration and that the design variables to be optimally adjusted are allowed to take on any real value in a given domain. To date, CONSOL-OPTCAD™ has been used on applications as different as design of various circuits, design of controllers for a flexible arm, a high-performance aircraft (rotorcraft control systems), a robotic manipulator, or determination of optimal flow rate and temperature profile for a copolymerization reactor [38].

VisualDOC has been developed by Vanderplaats Research and Development (VRD), Inc. mainly for the computer aided analysis community. A graphical user interface increases the flexibility of process integration, system automation, and design optimization. VisualDOC is a multidisciplinary design, optimization, and process integration software which can be used to define, execute, and automate a design process. It includes design modules such as Optimization, Design of Experiments, Response Surface Models, and Probabilistic (Robust and Reliability-based) Analysis which it can add to almost any analysis program. VisualDOC's graphical user interface allows the user to easily create a connected work-flow of components and configure them. VisualDOC supports multi-level, cyclic, and conditional workflows. Its features include comprehensive concurrent monitoring and visualization tools, storage and reuse of generated simulation data for post-processing, full debugging support for model execution, and the ability to interactively inspect and monitor the design process. It also supports remote execution in a heterogeneous environment, partial and batch-mode execution, and provides programmatic access to all the included design modules. It can integrate with Excel, Matlab, various CAE software, and user-defined libraries and executables. VisualDOC provides a useful and flexible Simulation Data Management (SDM) capability as can also be found

in other PIDO software. Many engineers and analysts spend an extraordinary amount of time engaged in parametric studies of one or two parameters using one or more different simulation programs. As other multidisciplinary tools, VisualDOC has been conceived to easily set up simulation(s) from user point of view, providing the basis for an automatic run and search of the best design while varying many parameters subject to many constraints [39].
ModeFRONTIER® is another example of integration platform for multi-objective and multi-disciplinary optimization. Such tool has been developed by Esteco with the final aim to manage complex engineering problems [40]. It provides a seamless coupling and interfacing with proprietary codes and third party analysis tools, enabling the automation of the design simulation process.
The final purpose of such tools can always be basically summarized with the capability to facilitate the process of analytic decision making. **Nexus** represents another optimization suite that has been developed by the iChrome Ltd. to provide useful capabilities in the context of multidisciplinary analyses. It allows the integration of design process, distribution and scheduling analyses, inter-operation and exchange data between applications, management, visualization and organization of results [41].
iSIGHT is one of the most widespread tools among the multidisciplinary analysis ones. It originated from the computing system for Computer Aided Optimization of General Electric (GE) and employs MDO Language (MDOL) as a unique operation script language. In this way the main objective is the capability to provide a customized environment depending on the architecture of problems and user's circumstances. Such desktop tool allows the interaction with CAD/CAM/CAE/PDM environments across different platform (Windows, LINUX or UNIX). Task Manager section of iSight is able to manage different optimization methodologies, including gradient methods, genetic algorithms, approximate methodologies and quality engineering methodologies. Parallel processing features are also available as well as the management of data flow and design strategies.
ModelCenter from Phoenix Integration currently play a key role within the market of the PIDO software. It mainly works on Windows O/S but it can construct heterogeneous distributed environments using the network for the connection with multiple analysis servers. In this way is can control different programs such as commercial CAD tools as well as analysis ones, supporting the JScript and VBScript to connect new entities. This features makes ModelCenter particularly flexible in cooperating with internet/intranet environments and databases. It relies both on a library of optimization methods and a large set of tool adapters for external interfaces.
A couple of interesting works for the MDO framework coming from academic and NASA activities are represented by Framework for Interdisciplinary Design and Optimization (FIDO) [42] and Intelligent Multidisciplinary Aircraft Generation Environment (IMAGE) [43]. In the first case the tool has been developed by NASA Langley while in the second one the framework has been implemented at Georgia Tech Aircraft System Design Lab (ASDL).
All these tools represent commercial PIDO solutions currently present on the market but open-source initiatives are also available. **DAKOTA** and **OpenMDAO** projects are two of the most promising, interesting and well known research initiatives. They provide useful functionalities for the management of multidisciplinary design problems without the cost limitations associated to commercial tools. The related benefits and drawbacks will be covered more extensively in the following sections. OpenMDAO has been developed more recently with respect to DAKOTA but has already highlighted some interesting capabilities as shown in [79].

4.2.1 Drawbacks of the current PIDO tools

Despite all the features and main advantages of PIDO software they are not the ideal solution for MDO environment. The PIDO tool concerns mainly optimization methods at the expense of data management and collaboration between users. Scenarios built from such infrastructure allow the individual user to perform complex surveys but limit the interaction among different users above all when the number of people involved in a project becomes large (i.e. in the advanced development phases of a system). In addition, also if a PIDO tool can handle different engineering software (i.e. CAD, CAE, etc.), it is not particularly effective and satisfactory environment for users from modeling perspective. A workflow management sys-

tem represents a fundamental element for the implementation of an effective collaborative infrastructure. PIDO tools are currently not well suited for the sharing of a wide range of data and resources. Alternative solutions implementing web-based technologies can better manage such kind of information. An infrastructure based on web services can more effectively combine analysis codes, optimization methods and data-base management system, enhancing the collaboration and reducing data consistency issues. Current PIDO tools are not as flexible as can be web-based application and often this feature makes the interface difficult to understand and learn with respect to a web environment.

4.3 OpenMDAO Framework

One interesting initiative in the field of MDO is represented by OpenMDAO project. The acronym MDAO in OpenMDAO definition stands for Multidisciplinary Design Analysis and Optimization, underlining as this framework has been conceived to face the problems linked to complex system design. This open-source framework has been written with Python code as this language offers many advantages in the context of simulations integration. In the current work such tool has been considered for the possible integration with the modeling framework since it shows some interesting features for the management and integration of simulation code. In the following lines a brief description of this framework is reported to show the promising capabilities that can be obtained through such infrastructure. More details are available from [44] and [81].

4.3.1 Mission

The main purpose of OpenMDAO research initiative is basically represented by the capability to integrate analyses coming from different sources under the same environment. In particular it allows to combine analysis tools (or simulation codes) from multiple disciplines, at different levels of fidelity, and to handle the interaction between them. OpenMDAO is basically defined to manage the dataflow and the workflow (that specifies which code is run and when in relation to the other ones) concurrently with optimization algorithms and other advanced solving methods. The current capabilities of OpenMDAO can be summarized in the following ones. It allows the information exchange among multiple analysis codes at various levels of fidelity to create simulations and models of complex systems. Such infrastructure provides also the state-of-the-art MDAO algorithms for solving highly coupled analyses. Such problems can potentially arise when multiple tools are combined and integrated between each other. The object oriented approach (enhanced also by the use of Python language) allows quick implementation of new tools and methods for the management of increasingly complex situations. A recent and detailed report about its usage is available from [78].

4.3.2 Elements and their functions

OpenMDAO is extremely flexible thanks to the separation between the flow of information (dataflow) from the process in which analyses are executed (workflow). Such distinction is achieved through the use of four specific constructs, represented by the following ones:

- Component
- Assembly
- Driver
- Workflow

The construction of the overall analysis scenario starts with wrapping or writing from scratch the various analysis codes. During such phase these elements are basically used to build the Components. They are basically the building blocks for the construction of more complex system and related analysis. In

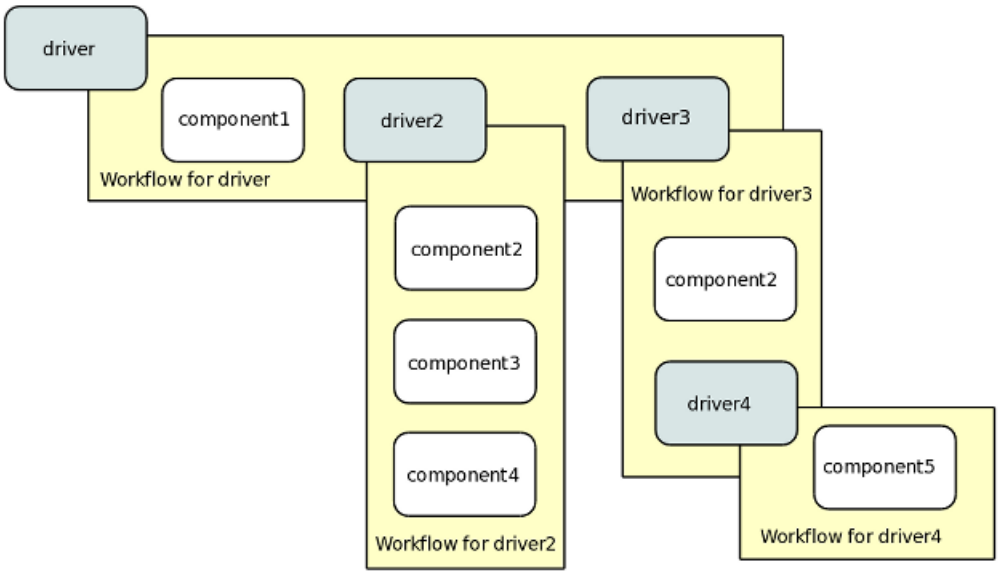

Figure 4.10: Overview of an example iteration hierarchy with few drivers [45].

particular the Python cross-platform capabilities are used to wrap the analyses that we want to integrate together, reducing in this way the efforts required to ensure data consistency, avoiding also the problems of O/S dependencies. Once a set of Components are available they are integrated to define an Assembly. In particular a group of Components is linked together within an Assembly to specify the dataflow between them. The following step is the set up of the workflow once the dataflow has been defined. In particular the procedure that drives the problem solving is affected by the Driver elements chosen in this phase. Drivers can be selected among optimizers, solvers, design of experiments, etc. Such information is used to basically define how to problem will be solved, scheduling the execution of the various analysis objects. To better explain the relationships between the dataflow definition and workflow definition it is possible to say that multiple Driver/Workflow combinations may exist for the same dataflow. For example the same dataflow can be used to run a straightforward optimization on the system, to develop a set of surrogate models first and then perform an optimization on the models or to run a sensitivity analysis. In this case the dataflow is the same but there are three different workflow with different purposes. An example of an iteration hierarchy involving different drivers is reported in figure 4.10 ([45]).

OpenMDAO is able to provide a wide set of features that make such framework useful to build complex analyses in the MDAO field. A much higher degree of code sharing, re-use and modularity is also achieved through a common platform, enhancing the data exchange among the MDAO community. Algorithms and solving methods can then be developed and distributed among users and communities, improving also the validation and investigation of new techniques. The main features provided by OpenMDAO are represented by a library of built-in solvers and optimizers, tools for meta-modeling, data recording capabilities, support for analytic derivatives, support for high-performance compute clusters and distributed computing, extensible plugin library. All such functionalities are available through an object oriented approach that enhances the integration among different environments. In figures 4.11 and 4.12 are shown two examples of the possible interactions among components within the same assembly as well as between assemblies on different levels.

The development effort of such framework is driven by the NASA Glenn Research Center, with also support of NASA Langley Research Center. NASA's interest in the OpenMDAO project comes from the evaluation of unconventional aircraft concepts like Turbo-Electric Distributed Propulsion. Although NASA's focus is on analyzing aerospace applications, the framework itself is extremely flexible and can used also in other disciplines [45].

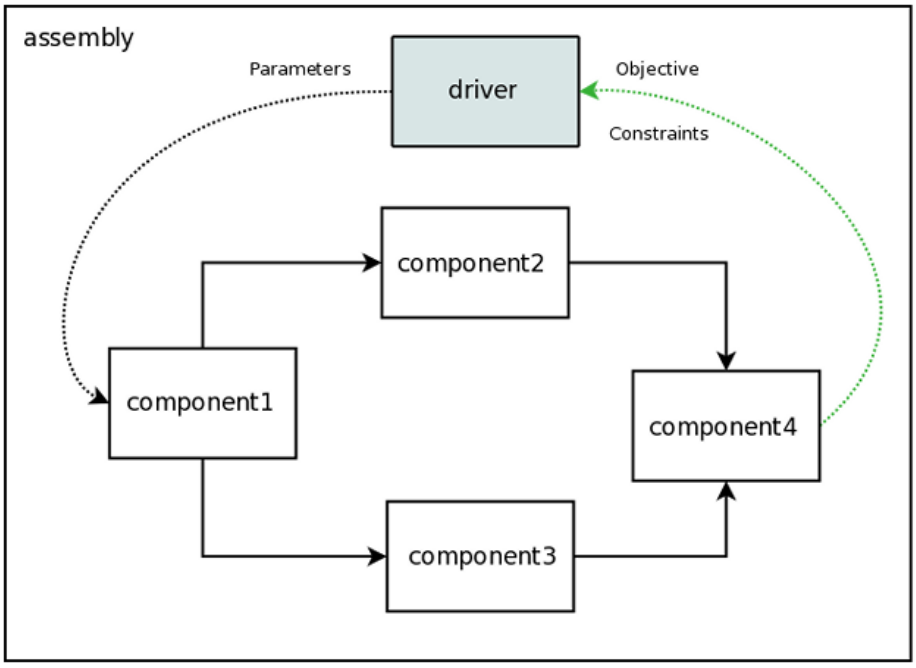

Figure 4.11: Data flow among components of the same assembly [45].

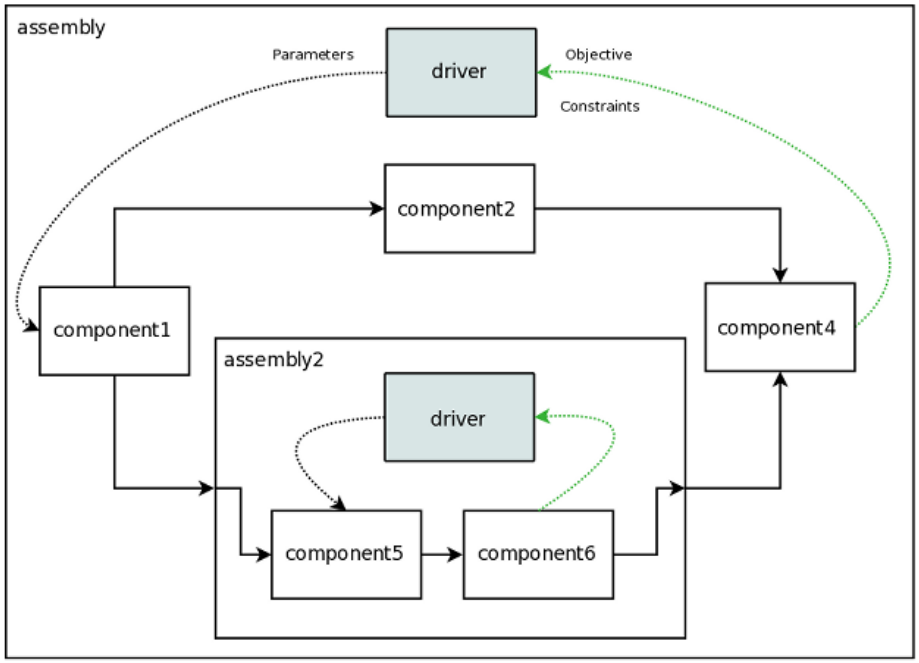

Figure 4.12: Interaction among different assemblies placed on different levels [45].

4.3.3 Browser GUI (Web Based)

In addition to the advantages previously described (integration capabilities with analysis environments, object oriented platform, open-source initiative, solving methods library, etc.) OpenMDAO has also started the integration with the current web-based technologies. In particular the tool provides a graphical user interface as a web-service running on locale machine. It is implemented basically as a local server lunched from the command line window and it can be accessed from the browser and with no necessarily a network connection (due to the fact that the server is launched on the same machine). Such interface can help to set up the dataflow pattern, providing also the information needed for the specification of the workflow. Such activities can be performed also thanks to the aid available of drop down menu and drag and drop elements, reducing the possibility to erroneously design incorrect connections or instantiation for example. The same approach can also be considered for future extension of the same framework, paving the way for the possible integration within a more complex web-based infrastructure where the multidisciplinary analyses are managed with the support of OpenMDAO (one of the possible services provided by such web-based infrastructure).

4.4 DAKOTA

DAKOTA project represents one the most interesting initiative that involves the definition of powerful open-source tool. In particular we have explored the feasibility to use this instrument for the management of the development process related to system implementation. In the context of a multidisciplinary design optimization study DAKOTA provides a series of useful instruments that allows controlling the simulation data. Generally speaking under the same environment it is possible to process and to transfer the information among different analysis tools. In this way we can consider the opportunity to automate some of the pre-processing and post-processing operations that traditionally are performed manually. Within an industrial perspective this allows to reduce the time to market feature, reducing consequently also the costs involved in the development process. The evaluated tool comprehends different key capabilities that we briefly present in the following lines (more details are available in [77]).
Historically DAKOTA born at Sandia National Laboratories as an instrument for the proper prediction, simulation management and risk-informed design process. The main goal is to provide all the useful information that may be necessary for the decision making phase that a system development generally requires. The simulation credibility that is pursued relies on various activities as those related to validation, to verification, to uncertainty qualification and finally to physics modeling fidelity. All this features depend in turn on other sub-activities that concur in the generation of non-deterministic results on which simulation credibility places its success.
DAKOTA includes a wide set of algorithm capabilities and utilities that allow to manage complex model simulations, as acronym highlights (i.e. Design and Analysis toolKit for Optimization and Terascale Applications). Basically it helps to understand which the relevant parameters that affect product behavior are and to establish how uncertainty influences the system responses. Other functionalities are related to optimization analysis and to calibration of analytical functions with experimental data. All the tool operations are based on interpretation of response metrics and generation of process parameters. In particular response metrics come from computational model (simulation) as the introduced parameters are required to set the simulations execution. The computational model may be black box or semi-intrusive software program. In the first case are included any codes describing mechanics, circuits, high energy physics, biology, chemistry, etc whereas in the second one we can find Matlab, ModelCenter, Python, SIERRA multi-physics, SALINAS, Xyce, etc. All these features may be implemented within an automated iterative analysis, supporting for example also experimental testing through computer models. In this way is possible to run many situations not well understood and then physically test only a reduced number of worst case scenarios. We briefly introduce also the main tool-kit features. It provides:

- Generic interface for external simulation.

- Time-tested and advanced algorithms to manage different type of variables (non-smooth, discontinuous, multi-modal, discrete and mixed).
- Strategies to combine methods and integrate different environment (advanced studies and surrogates generation).
- Capability to address mixed deterministic and probabilistic analysis.
- Possibility to address the execution of simulations cycle on clusters through scalable parallel computations.
- Advantages of an object oriented code.

4.4.1 Sensitivity Analysis capabilities

Sensitivity analysis is one of the main features that DAKOTA helps to manage. The final aim of this analysis is to understand how code outputs vary depending on changes in code inputs. In particular the variations on outputs are traced to the input perturbations through the use of automated process. Local sensitivities are generally managed with numerical implementation of partial derivatives while global ones are found via sampling methods and regression approaches. At the end of the activity the primary purpose is to identify which variables have main influence on the simulation results, allowing a more efficient running of the optimization or uncertainty quantification processes.
Generally these methods may be integrated within parameter study, design and analysis of computer experiments. In more detail the general sampling techniques implemented are the following:

- Single and multi-parameter studies (grid, vector, centered).
- DDACE (grid, sampling, orthogonal arrays, Box-Behnken, CCD).
- FSUDACE (Quasi-MC, CVT).
- PSUADE (Morris designs).
- Monte Carlo, Latin hypercube sampling (with correlation or variance analysis, including variance-based decomposition).
- Mean-value with importance factors.

The final responses related to sensitivity analysis are basic statistics, including mean standard deviation and possible correlations between the considered input variables. All these information may be collected in a tabular output that can be processed with an external third-party statistics tool.
The main effects and interactions between the variables are not generated through input distribution assumption for this kind of analysis.

4.4.2 Parameter Study capabilities

One of the main features of DAKOTA is represented by the Parameter Study Capability. In this case the effect of parametric changes through simulation models are shown on output responses. The input selected points in the parameter space are used to evaluate this particular type of sensitivity analysis. The input data sets can be selected in a deterministic way and structured with a particular pattern also if it is possible to use user-specified data group. There is also the possibility to use four different parameter study methods, introduced in the following list:

- Vector
- List

- Centered
- Multidimensional

These methods are different from each other on the basis of the techniques that are used to identify the parameter space points. In the case of Vector method the parameter study is performed choosing the input design points included on the vector line between two points of an n-dimensional space on the basis of a selected number of intervals to be sampled. This approach encompasses both single-coordinate parameter studies as well as multiple coordinate vector studies. In the case of list methods the user supplies the list of input parameter number to be used in the study. The centered approach considers an initial point of n-dimensional space from which the other points are evaluated moving along the coordinates axes in different possible manner. This capability could be used for post-optimality analysis verifying that the identified solution is actually at a minimum or constraint boundary and also in analyzing the shape of the stationary point under consideration. In the final methods a hypergrid in n-dimensional space is created and the user has only to specify the number of interval to consider. This study generally is not used to link the response data set to any specific results interpretation but may be used as starting point for sensitivity analysis. In particular the response data set can be integrated with the evaluation of numerical information related to the gradients or hessian quantities. The parameter study can also be used to evaluate the nonsmoothness in the simulation response variations, refining model characteristics or setting the step size for the computation of numerical gradients. This capability can also be used to investigate problem area in the parameter space as also to perform simulation code verification, identifying the possible problem related to simulation robustness. The results coming from this analysis can be used as starting point for minimization methods as either a pre-processor utility. The same approach is used in the case of post-processing activities for example for post-optimality analysis. Parameter study settings require the definition of initial point and bounds for the design variables (or equivalently the initial state and bounds in the case of state variable) to proper manage the simulation run over the variables range. Parameter studies, classical design of experiments activities (DOE), design/analysis of computer experiments (DACE) and sampling methods have all the same main objective that is the proper exploration of parameter space. In particular the parameter studies are generally used for simple studies with repetitive structure. Vector or centered methods are addressed to local sensitivity analysis or assessment of function smoothness. Multidimensional technique is often used for the generation of grid points and plotting of the response surfaces.

4.4.3 Design of Experiments capabilities

One of the other capabilities of DAKOTA toolkit is represented by the Design of Experiments feature (DOE). In particular the classical DOE and the more recent design and analysis computer experiments (DACE) methods are both techniques that try to extract as much information from a parameters space as possible with a limited number of set points. DOE techniques are usually employed in the case of technical domains characterized by some randomness and nonrepeatibility of the experiments (for example in agricultural or experimental chemistry fields). In this way the main aim is to distinguish between the simulated (computer) experiments and physical experiments. The last one is characterized by a greater stochastic component that drives to the consequences that the same treatment may results in different outcomes. In computer simulation experiments instead there is quite often a deterministic code. Central Composite Design, Box-Behnken Design, Full/Fractional Factorial Design are some of the techniques included within the DOE classical approach. These ones allow extracting important information starting from set points mainly placed at the extremes of the design space, since this location offer reliable behavior in the presence of nonrepeatibility. The nonrepeatibility component in the case of computer simulations is the main characteristic that allows distinguishing between DOE techniques and DACE methods. Orthogonal Array Sampling and Latin Hypercube Sampling are in the case of DACE approach the more commonly used for the extraction of proper trend responses from simulation models. Other sampling techniques as Quasi-Monte Carlo approach are employed in DACE methods to uniformly cover the unit hypercube of design space. Generally speaking DOE/DACE techniques use only the results coming from the input parameters

bounds to construct the set of points for the extraction of required information. From this viewpoint these methods are substantially particular examples of the more general probabilistic sampling for uncertainty quantification. They are used to investigate overall simulation results, identifying the main effect of input parameters. This information is potentially employed to build the response function/surface for the following optimization/trade-off algorithm.
DAKOTA toolkit offers several packages for the management of activities linked to DOE/DACE processes. Some of the main important ones are introduced in the following list:

- Latin Hypercube Sampling (LHS) package
- Distributed Design and Analysis for Computer Experiments (DDACE)
- Florida State University design and Analysis of Computer Experiments (FSUDACE)
- Problem Solving Environment for Uncertainty Analysis and Design Exploration (PSUADE)

Some of the mathematical methods implemented within the named package can be briefly summarized in the following list:

- Orthogonal Array
- Box-Behnken Design
- Central Composite Design
- Random Design
- Quasi-Monte Carlo Sampling based on Halton or Hammersley sequences
- Centroidal Voronoi Tessellation
- Morris Screening

4.4.4 Uncertainty Quantification capabilities

One of the other important DAKOTA feature is represented by the Uncertainty Quantification. The main purpose of this type of analysis is to understand how an assumed distribution for the input variables is propagated on a distribution for the output response. In this case forward propagation is considered with the aim to quantify the non-deterministic effects on model output. The related methodologies allow managing probabilities of failure (reliability metrics), robust optima and also quantification of uncertainty when calibrated models are used for behavior prediction. The uncertainty quantification methods can also exploit the results coming from multifidelity modeling to drive complex analysis. On this topic a clear description is available in [76].
Dakota toolkit provides useful instruments for the evaluation and characterization of epistemic uncertainties and aleatory uncertainties. Simulation models are often affected by the presence of phenomena that do not show a deterministic behavior. In particular some simulation parameters such as material properties or boundary elements (e.g. phenomena external to the system under consideration) are characterized by uncertain values. These quantities are modeled through the use of probability distributions that describe the element response over a particular range of values. The right evaluation of these values is fundamental for understanding the potential ranges of outputs or scenario implications. The capability to evaluate the effect of uncertainty is particularly relevant in the decision making process. Uncertainty evaluation is generally differentiated between two main categories: the epistemic uncertainty and aleatory uncertainty.

- Epistemic uncertainty
- Aleatory uncertainty

Epistemic uncertainty represents the uncertainty related to the lack of knowledge of a particular quantity and it is often expressed equivalently as state of knowledge uncertainty, subjective uncertainty, type B uncertainty or reducible uncertainty [75]. Generally speaking this type refers to the cases where uncertainty can be reduced through increased understanding or increased and more detailed data. Interesting analyses about such topic are available from [73] and [74]. In particular epistemic quantities are referred to that elements which have a fixed value in an analysis but we do not known that fixed value. For example, the elastic modulus of a material in a specific component is generally fixed but unknown or poorly known. On the other side the aleatory uncertainty is characterized by relative randomness which cannot be reduced by further data collection. For example the uncertainties related to weather cannot be reduced by gathering further information. Aleatory uncertainty is also expressed as stochastic, variability, irreducible and type A uncertainty. Aleatory quantities are usually defined with probability distributions when epistemic cannot be modeled in the same way. There are many ways of representing epistemic uncertainty as for example the probability theory, fuzzy sets, possibility theory and imprecise probability. The right choice between these alternatives represents a challenging research topic. Three of the most widespread way of epistemic uncertainty evaluation are represented by the interval analysis, Dempster-Shafer evidence theory and second order probability. The last solution is often used in the case of mixed aleatory/epistemic uncertainties. In the case of interval analysis it is assumed that nothing is known about the uncertain variables except that they lie within certain intervals. In this case the main aim is to identify the range of values within which the output values will lie. In Dempster-Shafer evidence theory the choice of input parameters is managed through the assignment of probability value to the different subranges with which the overall variables range is divided (Basic Probability Assignment). Finally the second order probability evaluation is based on the management of both the aleatory and epistemic uncertainty. An example of such an application is represented by the case where the probability distribution type is known (e.g. that it is distributed normally or lognormally) but the parameters governing the distribution is not well known. This situation is faced through the use of an outer and an inner loop. The outer loop manages the choice of epistemic values of the related governing parameters for the considered distribution while the inner loop is characterized by the sampling. This last process is performed from the aleatory distribution with distribution parameters set on the outer loop (on epistemic point of view).
Different techniques are used to propagate the aleatory behavior related to the probability distribution:

- Latin Hypercube Sampling
- Local Reliability Method (mean value, MPP search, FORM, SORM)
- Global reliability methods (EGRA)
- Non-intrusive stochastic expansion methods (polynomial chaos and stochastic collocation)

The just considered methods are used to face aleatory uncertainty while the epistemic one is managed with the following ones:

- Local/global interval estimation
- Local/global Dempster/Shafer evidence theory (belief/plausibility)
- “Second-order” probability

In particular DAKOTA can output probability of response thresholds, reliability metrics, response corresponding to a metric, etc... In this case the so defined “Second-order” probability refers generally to the nested sampling technique that are frequently used in Quantification of Margins and Uncertainties (QMU). Uncertainty quantification (UQ) is one of the main instruments that allow a better understanding the behavior of a particular system. DAKOTA toolkit offers a wide range of uncertainty quantification instruments for the management of information about the available data. Nondeterministic analysis is addressed to the characterization of the uncertainties on model inputs and their influence on outputs through computational simulation. In DAKOTA the uncertainty quantification is mainly focused on the forward propagation

of the process, involving the statistic generation of outputs distribution. UQ is particularly linked with sensitivity analysis since in both case the main aim is to understand how variations in the inputs values affect outputs probabilistic distribution. Generally speaking the output stochastic distributions are inferred on the basis of the assumed input ones. As previously introduced the uncertainty quantification can be distinguished between the aleatory and epistemic variability. The considered toolkit offers a series of functionalities that allow managing both these uncertainty types. The main aleatory uncertainty quantification methods used within DAKOTA are introduced in the following list:

- Sampling-based approaches
  - Monte Carlo
  - Latin Hypercube
- Local Reliability method
- Global Reliability method
- Stochastic Expansion
  - Polynomial Chaos Expansions
  - Stochastic Collocation

The epistemic uncertainty evaluation methods are instead listed in the following:

- Local Interval Analysis
- Global Interval Analysis
- Dempster-Shafer Evidence theory

In the case of mixed aleatory/epistemic uncertainty quantification DAKOTA supports the following methods:

- Interval-valued probability
- Second Order probability
- Dempster Shafer theory of evidence

The Latin Hypercube package provides both Monte Carlo random sampling method and the effective Latin hypercube approach. The probabilistic distributions that can be considered within the evaluated toolkit are: normal, lognormal, uniform, loguniform, triangular, exponential, beta, gamma, gumbel, frechet, weibull, poisson, binomial, negative binomial, geometric, hypergeometric and finally user-supplied histograms. The uncertainty quantification process can be realized also with the possibility to use a user provided correlation matrix. In this case the correlations between input and output variables are deduced from the information available and not from the simulation results. The incremental Latin hypercube sampling is sampling method based on the increase of the sampled points between two consecutive extraction operations and carrying the information gathered from the previous one. The reliability methods implemented in DAKOTA can be applied in some cases with different alternative modes on the basis of the type of the level mappings. Some techniques solve local optimization problem to find the most probable point for a particular quantities and then about this one the probabilities approximations are integrated. Some of the techniques are reported in the following list:

- Mean Value (MV) method
  - First order version (MVFOSM)

  - Second order version (MVSOSM)

- Most Probable Point (MPP) search method (forward Reliability Index Approach (RIA) mode or inverse Performances Measure Approach (PMA) mode)
  - Advanced Mean Value method (AMV)
  - Iterated Advance Mean Value method (AMV+)
  - Two-point Adaptive Nonlinearity Approximation method (TANA)
  - First Order Reliability Method (FORM)
  - Second Order Reliability Method (SORM)

The stochastic expansion methods employ the use of projection, orthogonality and weak convergence to evaluate the related statistics. Polynomial Chaos Expansion (PCS) uses multivariate orthogonal polynomials which is particularly suited for the representation of a defined input probability distribution. Stochastic collocation instead employs multivariate interpolation polynomials. The evaluation of expansion coefficient in the case of PCE can be done with the following techniques for numerical integration:

- Spectral Projection approach
  - Sampling
  - Tensor-product Quadrature
  - Smolyak Sparse Grid
  - Cubature method
- Regression approach
  - Least Squares
  - Compressive Sensing

Stochastic collocation interpolants can be formed with the list reported in the following:

- Tensor-product
- Sparse Grid

The interpolants can be expressed under different combination:

- Local or Global
- Value-based or Gradient-enhanced
- Nodal or Hierarchical

The Importance Sampling method is more effective than Monte Carlo sampling and is generally used for failure probabilities computation. In this case the sampled points are generated in preferential regions of the parameter space often near the failure area for example or however defined by the user. Adaptive sampling technique tries to build a surrogate model that allows reducing the computation loads related to a more complex simulation. A first set of sampled points is chosen (for example with Latin hypercube methods) and then the related grid is adaptively modified and updated on the basis of selection criteria. Interval analysis is mainly used in the context of epistemic uncertainty evaluation and the local or global techniques implemented are addressed to the identification of the output bounds on the basis of input ones. In the case of global approach optimization methods (based in particular on Gaussian process surrogate model) or sampling techniques are used to assess bounds. The local methods use instead gradient information through Sequential Quadratic Programming (SQP) or Non-linear Interior Point (NIP) to obtain bounds. The

Dempster-Shafer Theory of Evidence is mainly used to model the effect of epistemic uncertainties, basing its implementation with the definition of basic probability assignments (BPA) to each interval for the design variables space. DAKOTA provides also other instruments as for example those related to the Bayesian Calibration. In this approach the uncertain parameters are defined through a previous distribution (assumed on the basis of the known characteristics of the modeled phenomena). This first distribution is then upgraded with experimental data and after the process of Bayesian Calibration a posterior distribution is obtained. Reliability methods represent an alternative way to evaluate uncertainty quantification with the aim to introduce a less computationally demanding with respect to sampling techniques. Starting from specified uncertain variable distributions the response function statistics are computed. The response statistics include mean, standard deviation, cumulative distribution functions (CDF) and complementary distribution functions (CCDF). The probability calculations involve often multi-dimensional integral over an irregularly shaped domain for the variables of interest. Under these conditions it may be very difficult to proper manage the information gathered and also to process the data available. For this reason often these techniques employ the definition of a variables transformation through the definition of mapping functions between two equivalent variables spaces where the final one is easier to monitor. In DAKOTA the implementation of this mapping is obtained through the use of Nataf transformation, which is similar to Rosenblatt transformation in the case of independent random variables. The global reliability methods are generally used to manage non-smooth and multimodal failure surfaces introducing a global approximations based on Gaussian process models. The technique implemented in DAKOTA is identified with the Efficient Global Reliability Analysis (EGRA) which belongs the family of Efficient Global Optimization (EGO) methods. In particular the approximation obtained is used to drive the search activity of the points that maximize the Expected Improvement Function (EIF). The exploration of design variables space proceeds to find the points that show a higher value of probability to represent better solution. Briefly speaking the optimization methods that reflect the EGO approach are characterized by the following steps:

- Definition of the initial Gaussian process model for the objective function.
- Search of the point that maximizes the EIF evaluation, stopping for those points that show a small EIF with respect to the previous evaluation.
- Evaluations of the objective function for those points that have highlighted an upper value of EIF. From this information in new value of the objective function in these new points the Gaussian approximation of the objective function is updated.
- The process is then repeated from the second step.

The main methods included within the Stochastic Expansion approach are represented by the polynomial chaos expansion and stochastic collocation. The polynomial chaos expansion is based on a multidimensional orthogonal polynomial approximation while the stochastic collocation is based on a multidimensional interpolation polynomial approximation. In both cases the approximation starts from the definition of standardized random variables. The feature that characterizes these two methodologies is represented by the fact that the final solution is expressed as a functional mapping and not only as a set of statistics as in the case of other nondeterministic methodologies. In particular DAKOTA implements the generalized PCE approach using the Wiener-Askey scheme where different orthogonal polynomials are used for the modeling of the effect of continuous random variables described by various probability distributions. The main difference between the stochastic expansion methods implemented (PCE and SC) is that, whereas PCE estimates coefficients for known multivariate orthogonal polynomial basis functions, SC employs multivariate interpolation polynomial bases for known coefficients. The interpolation polynomials can be local or global and also value-based or gradient-enhanced. The related four combinations are referred to Hermite, Lagrange, piecewise linear spline and piecewise cubic spline. In the case of global methods the sensitivity of the variables is evaluated through the use of the Sobol indices. In the case of Stochastic Collocation it is possible to follow different procedures to evaluate the orthogonal polynomials which can be generated from Gauss-Wigert recursion coefficients in combination with the Golub-Welsch procedures for example.

The main goal of adaptive simulation is to build a surrogate model that can be unused in place of a more expensive simulation model. The adaptive simulation model can be implemented following the next step:

- Evaluation of the expensive simulation (considered as the true model for the phenomena under analysis) at the initial sample points.
- Fit/refit of a surrogate model.
- Creation of a candidate set and score based on the information gathered from the surrogate.
- Selection of new candidate points to evaluate again the true model (the more expensive one).

The evaluation of the score list for the selection of the candidate points is base on different types of metrics. DAKOTA implements the following ones:

- Predicted Variance
- Distance
- Gradient

Once the score list has been defined the choice of the identified points can follow different approach on the basis of various methodologies. In fact once the set of points has been ordered the choice of the points to update the approximation can take count also of the position of this points along the design variables space (for example of two point with high score but near each other in the design space it should be better to select only one of them). On the basis of this consideration in DAKOTA different choice strategies has been implemented:

- Naive Selection
- Distance Penalized Re-weighted Scoring
- Topological Maxima of Scoring Function
- Constant Liar

In Dempster-Shafer theory of evidence the ranges of variables are defined through the terms of belief and plausibility. These functions allow evaluating the statistical functions related to a particular simulation response. The cumulative belief function is the lower bound on a probability estimate that is consistent with the evidence while the cumulative plausibility function is the upper limit that is consistent with the evidence. Considering again the Bayesian Calibration methods DAKOTA introduces the Markov Chain Monte Carlo (MCMC) as the standard technique used to compute the posterior parameter densities, starting from the given experimental/observational data. In particular the variation algorithm used within this framework is called DRAm which stands for Delayed Rejection and Adaptive Metropolis, also if other algorithms typologies can be implemented and they are however current research area. The DAKOTA implementations of Bayesian calibration follow two alternatives, one called QUESO and the other one GPMSA. QUESO stands for Quantification of Uncertainty for Estimation, Simulation and Optimization. The choice for the uncertainty quantification method to use depends mainly on the characteristics of uncertainties of the input parameters, the available computational budget and also on the objective accuracy to be obtained. In particular the once the class of method has been selected (choosing among sampling, local reliability, global reliability, etc.) the applicable methods (LHS, Monte Carlo, TANA, etc.) depends on the desired problem features.

4.4.5 Optimization capabilities

The optimization capabilities provided by DAKOTA can be recognized in the set of advanced algorithms available as for example those that allow managing multi-objective optimization as to perform surrogate-based minimization. In The optimization problem formulation design variables and design parameters stand for the same quantities. They belong to the design space also called parameter space while the terms design point or sample point refer to a particular set of values within the parameter space. The objective function denotes the simulation response that is monitored to manage the design variables choice. The constraints elements can be defined as linear or non-linear and also can be distinguished between equality and inequality behavior. The feasible and infeasible design points are defined with respect to the violation or not of the constraints spaces. The optimization capabilities can be analyzed on the basis of optimization problem type, search goal and search method. The optimization problem type categorization is based on the level of complexity that arises from the constraints and objective functions. From a hierarchical point of view the constraint categorization can follow the increasing complexity order, starting from simple bound constraints through linear constraints to full nonlinear constraints. In particular this division can be reported in the following list with increasing complexity order referred to the constraints type:

- Unconstrained problem: problem with no constraints
- Bound-constrained problem: problem has only lower and upper bounds on the design parameters.
- Linearly-constrained problem: problem has both linear and bound constraints.
- Nonlinearly-constrained problem: this problem can include the complete range of nonlinear, linear and bound constraints.
- Equality-constrained problem: when all the linear and nonlinear constraints are equality constraints.
- Inequality-constrained problem: when all the linear and nonlinear constraints are inequality constraints.

Another categorization can be made on the basis of the linearity of the objective and constraints functions:

- Linear Programming Problem (LP): a problem where objective function and all the constraints are linear.
- Nonlinear Programming Problem (NLP): a problem where at least some of the objective and constraint functions are nonlinear.

The search goal refers to the main aim of the optimization algorithm. In particular two different approaches can be considered:

- Global Optimization
- Local Optimization

In the case of global optimization approach the goal is to find the optimal solution over the all design space. In the case of local optimization the goal is instead to find the optimal value in a limited/restricted region of the design space. The choice between these two alternatives depends on the available computational budget as on the complexity of the simulation code considered. The search method topic refer to the implementation of the strategies used to find the new design point with improved objective function on the basis of the previous computation. In particular the search method can consider two main distinctions:

- Gradient-based method
- Nongradient-based method

In the gradient-based method the gradients information related to the response functions are used to locate the direction of improvement for the next design point. In this case the computation of the gradient information can be expensive and often not particularly accurate. In this situation and also in those cases that show nonlinear behavior the nongradient-based algorithms represent the better choice. The nongradient-based optimization includes numerous approaches and some of the most widespread are reported in the following list:

- Pattern Search methods: methods that belong to nongradient-based local techniques.
- Genetic Algorithms: methods that belong to nongradient-based global techniques.

Another class of optimization methods refers to the Surrogate-based optimization (SBO) family. The main target of these techniques is to reduce the number of actual simulation runs through the construction of a surrogate model on a limited set of function evaluation. Surrogates models can be managed in different manner:

- Local surrogates
- Multipoint surrogates
- Global surrogates
- Hierarchical surrogates

On the basis of the optimization problem different types of methods can be used. A list of the possible optimization methods categorized about the various families is reported in the following:

- Gradient-Based Local Methods:
  - Conjugate Gradient
    - Fletcher-Reeves Conjugate Gradient variant
    - Polak-Ribiere Conjugate Gradient variant
  - Sequential Quadratic Programming (SQP)
  - Newton Methods
  - Method of Feasible Directions (MFD)
- Derivative-Free Local Methods:
  - Pattern Search
    - Asynchronous Parallel Pattern Search (APPS) variant
    - Coliny Pattern Search variant
  - Simplex
    - Parallel Direct Search Method
    - Constrained Optimization BY Linear Approximations (COBYLA)
  - Greedy Search Heuristic
    - Solis-Wets method
- Derivative-Free Global Methods:
  - Evolutionary Algorithm (EA)
  - Division Rectangles (DIRECT)

This classification represents the main subdivision of the available optimization methods classes. Other additional optimization capabilities are represented by the multiobjective optimization, scaling and solvers in shared libraries. Recent optimization approaches are represented by the following ones:

- Multilevel Hybrid Optimization
- Multistart Local Optimization
- Pareto-Set Optimization

4.4.6 Optimization usage

The selection of the optimization methods available from DAKOTA must follow some considerations about the problem features. In particular the usage guidelines depend mainly on the type of variables in the problem (continuous, discrete and mixed), the search typologies (it is important to understand if the global search is needed or if the local is sufficient) and the constraints characteristics (unconstrained, bound constrained or generally constrained). In the same manner other important evaluations depend on the efficiency of convergence to an optimum (for example defined by the convergence rate) and the robustness of the method in the case of the design space (as expressed by the nonsmoothness).
The main distinction that can be done about the choice of the methods can be addressed on Gradient-based, Nongradient-based and Surrogate-based. The Gradient-based methods are highly efficient optimization methods with the best convergence among the other techniques. In the case where the simulation code provides the analytic gradient and hessian information the application of Newton method can allow reaching the quadratic convergence near the solution. In the case where the only gradient information are provided the hessian ones are computed from the storing of gradient data over the simulation output and superlinear rate of convergence can be obtained. This method is particularly suited for smooth, unimodal and well-behaved problems. In other case this method may however be applied but with more accuracy in the gradient search direction and bad results can be reached. Under these conditions multiple minima will be missed. For the management of gradient accuracy the analytical functions often are not available and in this case the numerical implementation is introduced. Forward differences or central-differences algorithms can be chosen on the basis of computational budget and gradient accuracy required (forward differencing generate more reliable data but with twice the expense with respect to central differencing). Nongradient-based optimization techniques are mainly introduced in the case of nonsmooth, multimodal and poorly behaved problems. The convergence rates that can be obtained in the search activity of the optimal design point are slower than those reachable with gradient-based algorithms. The computational cost of the implemented algorithms is greater than gradient-based methods since the number of function evaluations is generally very high. Nongradient-base approaches are often more robust with respect to the previously introduced category and can be easily integrated in a parallel computation schemes (exploiting the possibility to implement multi-core computations). Surrogate-base methods try to improve the effectiveness of optimization algorithms and least squares methods with the use of surrogate models. The use of surrogate models allow to smooth poorly behaved problems reducing the discontinuous response results that can be obtained from nonlinear simulations. The data fit applied on the simulation results coming from complex models allow exploiting the benefit of gradient-based algorithms, improving in this way the convergence rate. Global search methods are then applied to properly explore the overall design space with reduced computational costs while gradient based methods are then used to efficiently converge towards the set of possible local solutions that are identified. A summary table that shows the link between the method classification, the desired problem characteristics and applicable algorithms are presented in table 4.2.

4.4.7 Models - DAKOTA

DAKOTA toolkit interface is mainly based on the definition of the characteristics of the models to be managed. Once the iterators (which definition refers to the methods set up for the simulation) are implemented the execution requires the connection with models. In particular this phase involves the mapping

Table 4.2: Methods classification and applicable algorithms [98].

Method Classification	**Desired Problem Characteristics**	**Applicable Methods**
Gradient-Based Local	smooth; continuous variables; no constraints	optpp_cg
Gradient-Based Local	smooth; continuous variables; bound constraints	dot_bfgs, dot_frcg, conmin_frcg
Gradient-Based Local	smooth; continuous variables; bound constraints, linear and nonlinear constraints	npsol_sqp, nlpql_sqp, dot_mmfd, dot_slp, dot_sqp, conmin_mfd, optpp_newton, optpp_q_newton, optpp_fd_newton, weighted sums (multiobjective), pareto_set strategy (multiobjective)
Gradient-Based Global	smooth; continuous variables; bound constraints, linear and nonlinear constraints	hybrid_strategy, multi_start strategy
Derivative-Free Local	nonsmooth; continuous variables; bound constraints	optpp_pds
Derivative-Free Local	nonsmooth; continuous variables; bound constraints, linear and nonlinear constraints	asynch_pattern_search, coliny_cobyla, coliny_pattern_search, coliny_solis_wets, surrogate_based_local
Gradient-Based Global	nonsmooth; continuous variables; bound constraints	ncsu_direct
Gradient-Based Global	nosmooth; continuous variables; bound constraints, linear and nonlinear constraints	coliny_direct, efficient_global, surrogate_based_global
Gradient-Based Global	nonsmooth; continuous variables, discrete variables; bound constraints, linear and nonlinear constraints	coliny_ea, soga, moga (multiobjective)

between the input variables and the responses that can be obtained from the simulation. There is also the possibility to define single interface with a single model or other more complex connection as in the case where sub-iterators and sub-models are considered. The main ways through which the recursion capabilities are implemented are represented by the following relationships:

- Nested relationship
- Layering relationship
- Recasting relationship

Using these components it is possible to implement and integrate more complex simulation architectures. In the case of nested relationship a sub-iterator component manage the execution of sub-model simulation. In the case of layered relationship instead the sub-iterators and sub-models are used only on periodic updating of the main model. Finally in the case of recast relationship the response functions coming from the simulation are used to define new problem formulation. At the end another model category can be defined for the particular case of surrogate model. Recast models are used in the case of variable and response scaling, transformations of uncertain variables and related response derivatives to employ standardized random variables, multi-objective optimization, merit functions and expected improvement/feasibility. As previously introduced the construction of surrogate models can take account for the various techniques available from DAKOTA packages. The related methods are reported for clarity in the following brief list:

- Taylor Series Expansion
- TANA-3
- Polynomial Regression
- Gaussian Process (GP) or Kriging Interpolation
  - Surfpack GP
  - Dakota GP
- Artificial Neural Networks (ANN)
- Multivariate Adaptive Regression Splines (MARS)
- Radial Basis Functions (RBF)
- Moving Least Squares (MLS)
- Multifidelity Surrogates
- Reduced Order Models

Surrogate model accuracy can be locally improved through the use of correction methods that consider the evaluation of the truth model on particular iterations. Additive corrections can be introduced also for the first and second order functions evaluation to correct respectively the gradient information and hessian data. Beta correction as first-order additive correction allows enforcing the convergence and consistency between the surrogate model and the high-fidelity one. The second-order corrections can be implemented through the use of analytic, finite-difference and quasi-Newton Hessian methods. The corrections introduced in the trust region can be defined both with additive and multiplicative approaches. All the presented techniques for surrogate models definition represent different procedures with which the surface of the approximated response function can be fitted. This process can be resumed with three phases. In the first part we have the selection of the set of the design points to be considered. Then for the selected points the true function is evaluated from the related simulation run. Finally the information

gathered is used to compute the unknown quantities, depending on the approach adopted for the generation of the surrogate model. For example from these information the polynomial coefficients, neural network weights and Kriging correlation factors can be estimated, allowing the characterization of the approximation assumed. Taylor series for example is well widespread in the definition of some model above all when a purely local approximation is needed. TANA-3 method is instead a multipoint approximation technique based on the two point exponential approximation. In particular this approach is characterized also by the use of Taylor series approximation for the management of intermediate variables. In this case the powers of these intermediate variables are identified to match the information coming from the current and the previous expansion point. DAKOTA toolkit offers also the possibility to manage the generation of polynomial regression model with linear, quadratic and cubic approach. Kriging and Gaussian processes are used mainly for the construction of spatial interpolation models. The ANN models are another family of surface fitting techniques that employs a stochastic layered perceptron (SLP) artificial neural network. In particular this method uses the neural network based on direct training approach proposed by Zimmermann [71]. The multivariate adaptive regression spline method is a surface fitting technique developed a Stanford University. In particular the parameter space is partitioned into sub-regions on which forward and backward regression methods are applied to create the response function for each sub-region. In this way each single domain is characterized by its own coefficients and parameters. These values are then used to build the response function extended over the entire parameter domain to create a smooth and continuous surface. This approach does not guarantee that the obtained response function pass through all the data points evaluated. MARS is however particularly suited for the management of nonparametric surface fitting of complex multimodal data trends. In the case of Radial Basis Functions the values modeled depends on the distance from a center point defined centroid and the approximation is build starting from a sum of discrete number of weighted radial basis functions. The shape related to the function chosen can be of various types but generally the shapes are Gaussian-like or splines-defined. The evaluation of functions weights are obtained through linear least squares solution. The moving least squares are represented by an evolution and a more specialized version of the linear regression models. In the case of linear regression approach the sum of squared residuals (where residuals are represented by the differences between the approximated models and the true one at a fixed number of points) is minimized. In the case of more specialized version of weighted residuals the differences is also weighted for the determination of the optimal coefficients governing the polynomial regression function. The moving least squares techniques are moreover a class of techniques where the weighted coefficients are moved or recalculated for every new point where the response prediction is recalculated. The Multifidelity Surrogates models belong to the family of hierarchy type approximations that are often also called multifidelity models, variables fidelity models or variable complexity models. In particular these approximated models are obtained through different ways as for example a coarser discretization, a reduced element order, looser convergence tolerances or omitted physics phenomena. The reduced order models are represented for example by techniques as Proper Orthogonal Decomposition (POD) often used in computational fluid dynamics (also known as principal components analysis or Karhunen-Loeve in other fields). Another example is represented by the Spectral Decomposition (also known as Modal Analysis) in structural dynamics. The approximated models are obtained through the use of reduced basis and projection of the original high dimensional space to a reduced one. Nested models are particularly used in the case where sub-iterators and sub-models are needed to perform a complex system evaluation. The sub-iteration generally accepts variables from an outer level, performs the sub-level analysis and computes a sub-level response that is then passed again to the higher level. The solutions provided by this approach can involve different classes of problems as listed in the following:

- Optimization within optimization (for hierarchical multidisciplinary optimization)
- Uncertainty quantification within uncertainty quantification (for second-order probability)
- Uncertainty quantification within optimization (for optimization under uncertainty)
- Optimization within uncertainty quantification (for uncertainty of optimal solutions)

4.4.8 Variables - DAKOTA

The variables definition within DAKOTA toolkit represents the feature through which it is possible to set the parameters that are then managed by the available methods. Depending on the method that iterate over the model bounded to the simulation the variables cover different meanings. In the case of optimization study the variables are modified at each iteration with the final aim to obtain an optimal design solution. In parameter study, sensitivity analysis and design of experiments the variables are related to the exploration of parameter space. Finally in uncertainty analysis the variables are instead associated to aleatory distribution in order to compute the related aleatory characterization of the response output functions. The considered framework provides the management of different variables types: design variables, uncertain variables and state variables. In particular another categorization is based on the nature of the variables domains, distinguishing between continuous and discrete variables domain. The discrete variables domain can also subdivided into discrete range, discrete integer set and discrete real set.

- Continuous range
- Discrete range
- Discrete set of integers
- Discrete set of reals

Often the nature of the variables range affects the choice of the algorithm or method to be implemented for the problem resolution. For example the presence of variables coming from continuous range allows the selection of gradient-based methods in the context of optimization study while parameters associated to discrete range cannot be directly related to the previous defined ones. In particular discrete design variables are often well suited for the management by a non-gradient based methods as for example the genetic algorithms. Discrete variables can be classified as categorical and non-categorical ones. The categorical ones represent the parameters that cannot be relaxed during the running of the solution method. For example this class is represented by the number of particular mechanical components that cannot assume values also only slightly different from the designed ones. When the discrete variables can assume values however slightly different from the designed ones the class is represented by the non-categorical typology. In this case the variables values can be relaxed during the execution of resolution method. For example the choice among a series of standard thickness for a particular component can be managed as non-categorical discrete range due to the fact that the variable under study can change slightly its value.

- Categorical discrete variables
- Non-categorical discrete variables

Since engineering problems are often related to the presence of a wide class of aleatory uncertainties DAKOTA offers different distribution for the representation of continuous aleatory uncertain variables and discrete aleatory uncertain variables. In the following list is reported the distribution that can be considered in the definition of aleatory design variables:

- Continuous Aleatory Uncertain Variables
  - Normal
  - Lognormal
  - Uniform
  - Log-uniform
  - Triangular
  - Exponential

  - Beta
  - Gamma
  - Gumbel
  - Frechet
  - Weibull
  - Histogram Bin
- Discrete Aleatory Uncertain Variables
  - Poisson
  - Binomial
  - Negative Binomial
  - Geometric
  - Hyper-geometric
  - Histogram Point

State variable identify those parameters that are not directly involved in the design process and under these conditions there is no need to map them through the simulation interface. These variables are represented essentially by those values that do not represent the value to be chose within the design process for a particular problem, They are however important because their values are necessary for the execution of the computational model. For example they can be represented by the convergence tolerances, time step controls or any quantity that is fundamental for the execution of the model but it is not relevant for the design process. In the same way as in the case of design variables they can be classified as continuous range, discrete range, discrete integer-valued set and discrete real-valued set. They affect the computational model but are not active from the solution algorithms and their modification can result in the changing of the simulation conditions (resulting then in different results). The management of mixed variables by iterator depends strongly on the iterative method related to that study. This choice affects the subset or views that characterize the variables data that are active during that iteration. The coexistence of variables of different types influences the management of the overall parameters available, since some variables are modifiable by certain methods while other not. The latter one needs to be mapped through the interface in an unmodified status. The process with which the active variables are established is fundamental for the determination of the derivatives that must be computed. Another important feature that needs to be specified in the variables definition block is represented by the domain type that can be categorized as mixed or relaxed. The simulation interface needed in the case of communication between DAKOTA framework and external simulation codes is required to exchange information through the file-system. In particular the system calls and forks are obtained through the implementation of reading and writing process of parameters and results files. Before the simulation invocation DAKOTA creates a parameters file where it is possible to find all the information needed for the cycle execution. The format of the related file is available both in standard and APREPRO type (APREPRO denotes a Unix-based operating system module). Within this file it is possible to specify the variables, the active set vector, the derivative variables vector and analysis components. Each row specifies the value and the tag with which the object are identified. This approach allows managing the dynamic memory allocation. The variables are listed in a precise ordered representation following the different typologies presented previously (continuous, discrete, etc...) and the reported tag are those used by DAKOTA as those specified in the user's DAKOTA input file. In analogous way active variables vector, derivative variables vector and analysis components are reported providing first the related identifiers. Data representation provided with APREPRO format is the same as that adopted in the standard format and all the ordering conditions are the same but represented with a slightly different construct. The use of this module representation is bounded mainly to the advantage of directly interface the APREPRO utilities. These ones allow the integration of pre- and post-processing activities, simplifying the model parametrization. APREPRO utility allows also mapping the

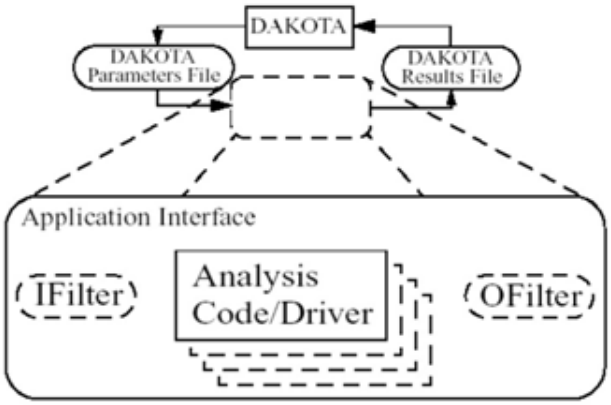

Figure 4.13: Components of the simulation interface [98].

variables passed through DAKOTA with that related to simulation code in a template file. In this way it is possible to populate directly the template file with the target variables. The introduced Active Set Vector represents a vector containing a set of integer codes and there are many integers as the number of the requested response functions for that study. In particular each response function has its own integer which identifies a well-defined set of requests for the corresponding function. Each integer is associated to a binary codification and a precise meaning. These latter refer to get hessian, gradient and value of the response function or get only gradient for example.

4.4.9 Interfaces - DAKOTA

The interfaces specification is one of the most important features that allow DAKOTA to manage simulations related to external codes. There are different types for the possible integration of the simulation interfaces between DAKOTA and external code simulation through the mapping of input and output parameters. One of those is represented by the integration of algebraic mappings. This one is represented by the possibility to bind the process management framework with code implemented in AMPL modeling language. In this case codes generated in AMPL can be used concurrently with AMPL solver library directly through DAKOTA. The required constraints are represented by the generation of particular files for the input and output parameters [46].
Simulations interfaces can be managed through three different approaches depending on the implementation of the code connection. The invocation of simulation codes can be realized with a system call, fork or a direct function invocation. In the case of system call and fork it is created a separated interface with respect to the DAKOTA framework and the related communications are realized through the exchange of parameter and response files. In the case of direct invocation a separated process is not created and the execution is realized within the DAKOTA process. The implementation of direct interface allows obtaining some advantages with respect to the computational demands. The code calls can be directly bounded to DAKOTA executable, avoiding the overhead linked the creation of input and output files for the code execution. This approach allows also the improvement of the performances related to the execution on parallel computers. The drawback is instead represented by the required conversion of existing simulation code into a library with a subroutine interface. The implementation of forks invocation is recommended with respect to the system call due to the portability and backward compatibility that can be assured. In the case of system call the invocation of system interface uses the system function from the standard C-library. The fork interface uses the fork, wait and exec families of functions to manage the simulation drivers. In particular a copy of DAKOTA process is created, replacing this copy with the simulation code or driver process. Transfer of variables and response data between management framework and simulation code are realized through the file system. The overall structure of the simulation interface components is represented in figure 4.13.

The elements reported in the representation can be identified in the system call invocation as in fork

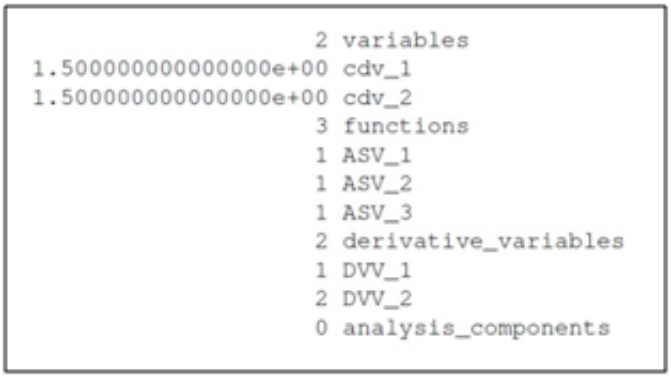

Figure 4.14: Standard parameters file format [98].

and direct integration. The input and output filters include the optional facilities that allow pre- and post-processing actions. These modules are needed to integrate the simulation code through the implementation of analysis driver while the input and output parameters are passed as command line arguments. In particular the related expressions that are directly connected with the function evaluation depend on the scripting language used to integrate the simulation code (UNIX C-shell, Bourne shell, Perl). In the case C/C++ program are used then the parameters files are passed as arguments through the traditional argc and argv instructions. The possible implementation that regards components interface depends mainly in the characteristics of the integrated analysis driver. Several solutions can be adopted depending on the needs.

- Single analysis driver without filters
- Single analysis driver with filters
- Multiple analysis driver without filters
- Multiple analysis driver with filters

In the case a single analysis driver is used and it is built to process directly the parameters and results files related to the framework then there is no need to implement filters. The files coming from DAKOTA are directly used by the simulation code to set the input and finally it generates the responses evaluation. System call and fork interfaces can be used to support asynchronous operation and also can be executed exploiting background running. The implementation of single analysis driver that filters invocation requires a different syntax with respect the previous integration. When multiple analysis drivers are involved in the simulation run the processes can be combined in a single system call through the use of syntax structure slightly different from that relative to the invocation of the single analysis driver. The interface process is supported by the features offered by the simulation file management that can help to debug potential errors. These features are represented by the file saving functionalities (before and after driver execution), file tagging evaluations, management of temporary files and work directories. The work directory set up is one of the most important features related to the invocation of DAKOTA cycle. In particular it is often convenient to execute filters and simulation codes in directory which is different from one where DAKOTA is launched. Moreover the evaluations of input and output files require that they are placed in separated directories to avoid potential conflicts between the various objects. The available data processing utilities allow the execution of various simulation codes in the proper way in order to manage the integrated simulation interface reducing the possibility of running problems. All the information that are needed for the definition of the parameter input file is generated from DAKOTA and then it is used by filters or other pre-processing files to set the simulation code execution (for example in the case where the proprietary codes require the generation of an input file following a well-defined template standard format). The related data are contained within the parameters input file following a certain data pattern. An example of such file is reported in figure 4.14.

Table 4.3: Active set vector integer codes.

Integer code	Binary representation	Meaning
7	111	Get Hessian, Gradient and Value
6	110	Get Hessian and Gradient
5	101	Get Hessian and Value
4	100	Get Hessian
3	011	Get Gradient and Value
2	010	Get Gradient
1	001	Get Value
0	000	No data required, function is inactive

The first line contains the integer representing the number of variables that are considered in the analysis. This row is identified with the tag "variables" that follows the integer number. Each line that immediately follows the first one represents a variable. The value associated to the considered variable for the current function evaluation is followed by the tag name that identifies it. Then the integer that represent the number of functions is introduced and followed by the tag "functions" that comes before the lines containing the functions set information. In particular those lines identify the active set vector information (ASV) and their tag. Then the integer representing the number of derivative variables and its tag "derivative_variables" anticipates the lines containing the derivative variables vector (DVV) and their tags. Finally it is possible to locate the integer for the number of analysis components (with the proper "analysis_components" tag) followed by the analysis components array and their tags. The descriptive tags for the variables are defined in this section and are mapped to the names expressed in the user's specification (if not provided by user names these tags are introduced as default descriptors). The Active Set Vector represents both the objective functions and constraints functions. For example if the considered problem has one objective function and two constraints then the ASV has dimension three. Along the list the objective functions come first and then the constraints evaluations. These ones are reported in the order consistent to the user definition. The derivative variables correspond to the variables introduced in the first vector specification. The information related to the “analysis_components” tags refer to the possibility of pass additional information for the simulation. In this way it is possible to provide at run time additional specifics that the simulation code can use to complete its execution (for example it is possible to pass to a structural code the name of a particular mesh file to be used). The same data structure can be highlighted in the representation related to the DPREPRO format parameters input file. Starting from the information contained within the parameters input file the user's supplied simulation interface must manage the process to proper generate the required format output. In particular on the basis of the previously introduced input file format the output file format have to satisfy the contained format directive. User's implemented interfaces have to access this input parameters file, process it, generate the input required for the external simulation code, execute this one and gather the created output. After this phase the results information has to be elaborated to create the output file that DAKOTA must process to proceed with the following cycle iteration. On the basis of the results for constraints and objective functions DAKOTA provides the new values for the input design parameters for the next input parameters file. The input parameters file previously shown contains the directive for the generation of data readable by DAKOTA. In particular starting from the Active Set Vector the output must contain in the same order a list of rows where each line contain first the value and then the tag of the related object represented (objective functions or constraints). The tags can be omitted due to the fact that the correspondence between the name of the result and its position along the list is uniquely determined. The rows containing the Active Set Vector integer data is important for the determination of the length of the information that will be contained in the output parameters file. In particular depending on that integer value it comes out the typology of data to be bounded with the object that is referred to. In particular the code that express the data to be included in the output parameters file (and that it is up to the user to recreate such file through the use of scripting module or filters files) are summarized in table 4.3.

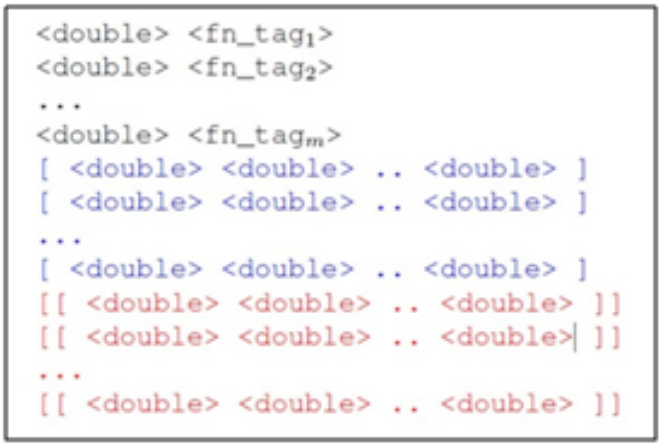

Figure 4.15: Results file data format [98].

On the basis of what entity is required from this specification integer number the corresponding simulation quantity (objective function or constraint) holds the computed values. For example if only the function evaluation values are required (integer = 1) then there are as many values as the simulation quantities.

4.4.10 Responses - DAKOTA

The section related to responses tag of the input parameters file manages the definition of responses specification. This part provides the pattern for the formats that the defined elements must follow. In particular these parameters include the definitions for the response functions as objective functions, constraints or calibration parameters. Also these specifications introduce the format representation of the first and second derivatives. On the basis of the considered techniques there are different response functions types that can be chosen. It is possible to define optimization data set, a calibration data set or a generic data set. Considering the availability of gradient information different types of gradient evaluation can be selected. In some cases gradient information are not needed and so gradients will not be used while in other context the required gradients data are numerically obtained through the finite differences approximation. In the cases the gradients will be supplied by simulation code the information gathered can be considered analytically. Finally it is possible to consider mixed gradients in the case the simulation code provides some gradient components while DAKOTA will approximate the remaining needed through finite differences. The hessian availability can be managed in the same way through different approaches on the basis of the available data. In some cases hessian information is not required from the iterative code while in other ones this information are computed numerically with finite differences applied over first-order differences of gradients or second-order differences of function values. In the case quasi-hessian specification is present then the required data are evaluated by a series of secant updates from gradient evaluations. In the case the hessian information are instead provided by the simulation code then the data required con be analytically elaborated. Mixed hessian are finally related to the cases where numerical, analytic or quasi-technique approaches are used. DAKOTA results file supports only one format expression while parameters-input file can be represented with two formats. An example of the results file pattern is reported in figure 4.15.

After the model simulation/iteration DAKOTA is expecting the generation of a file containing the information related to the output values with the format represented in the previous introduced figure. In particular the values provided must follow the function requests defined in the active set vector specification. In this file it is generally possible to enhance three different sections. The first section contains the function values and each row includes the related numeric evaluation and the tag that can also be optional. DAKOTA follows the order defined in the active set vector and the presence or not of tag string do not affect the correct evaluation of the contained information. The second block represents the gradients information which are provided within brackets and they must not be identified with any tag elements. In the same manner the hessian data are reported within double brackets and also in this case they do not be

identified by tag strings. The correspondence between the derivatives information and the variables to be used is provided by the Derivative Variables Vector (DVV). This vector contains the data needed to compute derivatives and the one to one correspondence is realized through the length of the vector itself. Inputs data of DAKOTA framework can be identified with two different formats. Annotated matrix and Free-form matrix represent the two potential alternatives that can be chosen for the data input exchange.

4.4.11 Outputs from DAKOTA

After DAKOTA iteration execution the results can be provided in different ways. In particular the outputs are reported on a text-based files that summarize the main event and iterator evaluation obtained from the simulations. During cycle running the data can be plotted on screen listing the information of the current execution while the same output can also be printed on a text file. In the same manner a text-file can be used to gather the process overall information (for example reporting the results of variables values, objective function and constraints for each iteration corresponding to a particular row of the file) in a tabular data file that can be easily post-processed with an external tool for visualization purposes. The standard output printed on the screen includes basically the evaluation number, the parameter values, the execution syntax, the active vector and the response data set. First an initial block reporting the main setting of DAKOTA process is plotted while the central part before the final block visualizes the results of each evaluation for the current iteration. For each function evaluation block are plotted the input variables values, the system call to the driver that manage the simulation driver and finally the results from the simulation execution (objective function and constraints evaluations). Before the evaluation of the objective function and constraint element is reported the active set vector. As previously indicated this object allows to express the types of data that are required from the simulator for the objective function and constraint element. In particular the integers contained code the evaluation of only function evaluations or also the gradient information, hessian etc... (the main guidelines that define this codification are included within the previous introduced section on the interface of the simulation code). Depending on the particular function evaluations settings the iterator can require for the single function evaluation additional computation related to the definition for example of gradient information. In this case for example each function evaluation can require the computation of gradient data through the estimation of other function (and so simulation) execution as for the finite differences approximation. This computations are like a sort of internal evaluations for the estimation of the data required to compute the direction of variables changes for that particular iteration. Finally the DAKOTA overall process is summarized in the final block where all the main information are summarized. In particular in this section it is possible to highlight the best values obtained for the optimization parameters, objective function, constraints, total evaluation counts and timing summary. Some other information can be plotted on the output stream on the screen depending on the characteristics of the implemented solution routine or coming from the features of the algorithm included in the solution library used. DAKOTA tabular format is also generated at the end of iteration process and the main purpose of such capability follows from the need to plot the obtained results on other external graphics plotting package. Some 2D visualization capabilities are available on UNIX platform while on Windows one they are not implemented. These features allow to plot the iteration results as the simulation run.

4.4.12 Examples applications of DAKOTA framework

In literature are present different examples of object oriented approach for the solution of engineering issues. Research activities have focused on addressing the challenges related to the application of iterative systems analyses to complex problems where simulations are expensive to evaluate and the response metrics may be poorly behaved (i.e., noisy, multimodal, discontinuous).
An example of such initiative is represented by the rSPQ++ Framework from the department of chemical engineering of Carnegie Mellon University [72]. This object-oriented tool for successive quadratic programming has been developed to support Successive Quadratic Programming (SQP) algorithms, allowing the integration with external specialized application. In particular different interfaces can be generated for

the connection with various linear algebra objects as matrices and linear solvers.
Examples of methods applied in the context of uncertainty quantification area are available from [120], [121] and [122], where the application of local and global reliability methods is investigated. In particular in [122] such techniques are applied for the study of microelectromechanical systems (MEMS). Other research activities regarding stochastic expansion or mixed aleatory-epistemic methods can be found in literature.
DAKOTA application to surrogate-based optimization is investigated in [123] and [124] where interesting results are provided. In the same way other surveys dealing with optimization and model calibration under uncertainty can be found in [125], [126] and [127].
Another challenging topic approached also with the use of DAKOTA framework is represented by the parallel processing and some results are available from [128] and [129].

Chapter 5

State of the Art

In the current chapter some of the most recent research initiatives regarding the integration of model-based approaches in the advanced phases of a project are briefly introduced. In particular a large number of examples that can be found in literature regard the integration of multidisciplinary design and analysis methods within a model based infrastructure. The management of alternatives and optional elements is not still properly considered from a model-based perspective. In the same way the integration of MDO techniques within a model-based environment is currently not well formalized, also if the design optimization across broad trade space is one of the main target capability of MBSE (as highlighted in figure 5.1). In the near future the cross domains analyses will be some of the most challenging activities that will characterize the development of MBSE "philosophies" and frameworks.

Different solutions may be considered for the actual implementation of MBSE paradigm to MDO design methods, as can be seen from reference literature for the same type of problem. Each solutions show advantages and drawbacks in relation to the specific context that has been considered and for this reason a unique, shared and comprehensive architecture is still far from being defined. The initial part of this work has been characterized by an analysis activity for the formal definition for the integration under evaluation. Various conceptual architectures has been preliminary proposed but only that one that seems to show a better behaviour has finally been considered and directly implemented in the framework under development.

5.1 Main problems and characteristics

Different kind of issues can arise when Multidisciplinary Design Optimization techniques are integrated with Model Based System Engineering methodologies. The development of the proper interfaces is strictly affected by the way such integration is actually implemented as well as it is placed within the overall design and analysis process. In the following sections some of the most important problems are briefly described to highlight the main aspects that must be taken into account when the overall infrastructure will include all such features. A clear understanding of the overall process and the related infrastructure must be properly defined to avoid the increase of issues when the actual implementation of the code is realized. If some concepts are not clearly well-posed during such phase then the issues can only increase in the following steps. A clear conceptual framework is then fundamental for the right evolution of the work, paving the way to the exploitation of the available resources.

5.1.1 Management of complex system

Currently the integration of complex aerospace systems requires the involvement of a large number of information. The amount of data stored, processed and exchanged is directly connected with the effectiveness of the overall process. The main conceptual infrastructure must be conceived to support the design and analysis process, trying to avoid negative consequences as data surplus for example. The management of complex systems is currently difficult to take under control as the number of variables is generally wide. In this case the correct handling of all such information represent one of the key-elements that affect the

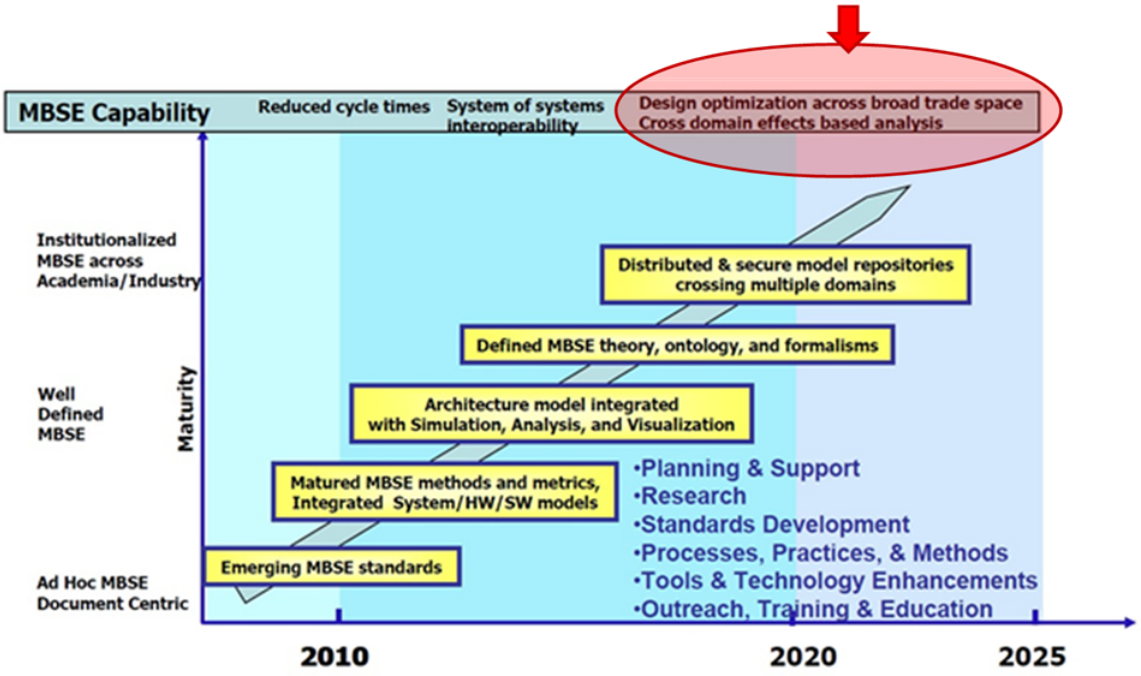

Figure 5.1: Design optimization capability highlighted on MBSE roadmap for the near future [14].

integration between MBSE and MDO. All data involved must be reflected in a well formalized conceptual infrastructure to ensure an effective connection between MBSE environments and MDO capabilities for example. In particular different strategies can be considered to manage project data since they often depend on various factors such as design processes structure, analysis workflows or used tools.
The effective management of system architectures is one of the most challenging activities and different solutions have been proposed and implemented by different research teams. The heterogeneity of all the subsystems and components that characterize the design phases of a product is an aspect difficult to take under control. From this point of view interesting results have been achieved by the Institute for Systems Research, University of Maryland. An overview of the related framework is available from [106] where a promising framework for model-based systems engineering is described. It consists basically by an integrated modeling hub and various application methods/tools which includes also tradeoff analyses via optimization.

5.1.2 Communication between domain-specific disciplines

Another problem that can affect the overall integration is represented by the communication between domain-specific disciplines. In this case the interfacing of MDO techniques with MBSE environment must foreseen the main features of the data exchange process between different domains. The communication among people with different backgrounds can widely affect the main purpose of the proposed approach. The use of different tools, procedures and format to model and analyze the same product must be properly coordinated. A shared conceptual infrastructure can widely improve the effective exploitation of MDO methods within an MBSE environment, ensuring the seamless exchange of data across the disciplines. In this way each discipline can however continue to use its own methods, tools and processes but the overall system model is shared on common basis. The communication of the data needed for a multidisciplinary analysis must be properly faced to ensure the connection with a model based methodology.

5.2 Possible solutions

The actual integration between MDO methods and MBSE methodologies can be approached in different manners, depending in particular on final objectives as well as the specific workflow that characterizes

the individual company. Multidisciplinary analyses can be used to investigate the product performances on the basis of the available set of data and the way such information is managed affects the integration architecture. Generally the solutions that can be adopted varies on the basis of the tools chosen as well as the main features for the platform to be considered. In particular it is assumed that the multidisciplinary analyses are managed by a dedicated platform, clearly distinguished from the simulation tools which provide the results managed by the platform itself (this distinction is based on the fact that currently some multidisciplinary analyses can also be executed within some simulation software while in this case the platform handling the overall cycle is more properly a process manager). From a conceptual point of view it is possible to identify two main possible alternatives that can be pursued for the final scope (but other ways can also be followed). In the first case the exploitation of MDO methods is based on the capabilities provided by an external multidisciplinary analysis platform. Such environment can be selected among the current available ones. In particular commercial solutions as open-source ones can be evaluated and considered for such integration. In this case a MBSE environment can be used to store all the representative information related to the system model while an actual external platform uses such data to set up multidisciplinary analyses. In this case the strategy foresees the implementation of all the adapters required for data exchanges among the involved simulation environments (the ones managed by the multidisciplinary platform). Data are collected, processed and then properly used to set up the chosen analysis (sensitivity analysis, optimization, uncertainty quantification, etc. for example). The exploitation of the MDO capabilities is basically provided to the MBSE framework as an external "service".
In the second approach the MDO capabilities can potentially be integrated within the MBSE environment, paving the way for a better use of the benefits deriving from a model-based philosophy. In particular the functions provided by a multidisciplinary analysis platform can be deeper integrate within the design process with respect the the previous solution. In this manner the it is possible to exploit the effectiveness provided by a model based infrastructure. For example the same optimization cycles can in fact be conceived directly within a model-based architecture (the same approach can be equivalently extended to the other analysis types such as sensitivity analysis, uncertainty quantification, etc.). In this second type, an object-oriented solution can widely enhance the advantages of a unique environment for the modeling and set up of multidisciplinary analyses, reducing for example the efforts required for the consistency check when data are exchanged directly with an external process manager (distinguishing for the sake of clarity such definition from the multidisciplinary environment which can potentially implemented also within a specific simulation environment with its own Domain Specific Languages - DSL). The process manager can then be conceived to be directly embedded within a system modeling framework, constituting a whole with the platform. This way provides promising capabilities but requires at the same time a clear understanding of the back-end structures and mechanisms from the implementation point of view. The overall interfacing for the involved simulation tools and environment can be pursued with less efforts if a common conceptual infrastructure is shared among the involved disciplines, reducing the time spent on consistency control. Such integration can be realized basically only if the platforms and the relate methodologies are clear and accessible. This situation often limits the choice to open-source initiatives and projects which ensure the possibility to directly manage the source code, customizing the already developed features to achieve the desired objectives.
In the following section some example of the current approaches is provided to show how such research topic is promising and that however different strategies can be pursued to assess the effectiveness of this integration.

5.3 Examples of research initiatives

Different initiatives have addressed their efforts towards the investigation of the potential benefits related to MBSE methodologies in the context of system design and analysis. The greater improvements have been mainly obtained within research centres or academic organizations. Many studies have been realized in this direction at the Jet Propulsion Laboratory (JPL). Similar analyses have involved academic institutions as the University of Michigan (System Engineering Department), California Institute of Tech-

nology (Caltech) and Massachusetts Institute of Technology (MIT). The majority of such surveys has been characterized by the investigation of SysML language actual benefits in the design and modeling processes from system perspective. The main applications regard the definition of architectural and behavioural architectures through SysML diagrams, considering in particular a representative model of the product. Only the last few years the research topics have started to evaluate the possible integration of such tools with external solver. In this way the main objective is to understand which advantages can be reached through such a methodology.
In the following sections some of the briefly introduced initiative will be considered with more details about the processes and approaches used, enhancing the characteristics that can be recognized among the various projects.
Some research activities also if not completely addressed to the evaluation of the complete set of MDO methods are however characterized by the assessment of feasibility of partial functionalities. For example some interesting initiatives are evaluating the integration of Sensitivity Analysis feature in the context of a model base framework.
Some interesting studies has in fact been done in the context of sensitivity analysis for the design process of space system in the context of model based system engineering environment [48]. In this case the main idea is represented by the implementation of a central integrated design environment which is then interfaced with different external tool (Excel workbooks, QUDV standards and Catia V5 script automation link). The developed framework follows the data model defined in the context of such study. In particular the data model foresees the presence of a system component that is related to zero or multiple parameters. The single parameter is then associated at least to one value. The system component also contains zero or multiple balancing where this element refers to the equation/relation that formalizes the relationships between the involved parameters (for example the physical relationship that characterizes the behaviour of the considered phenomena). The balancing element contains at least one or more source parameters and they represent the variable that formal cover the right hand side of the equation/relation (in particular they refer to the variable that are known and that allow to compute the quantity on the left hand side of the equation). The right/left side classification is not binding but allows to better express the fact that some variables are available (defined as source parameters) while one is computed explicitly (defined as target variable). This approach considers only one variable as the computed one. This pattern then defines two association towards the parameter class. One defines the source relationships (at least one or more parameters) and the other one represents the target relationship (only one parameter involved).
In this case the sensitivity analysis process can be considered as an interesting instrument in the context of the overall design and analysis process. Starting from the design activity the following phase is represented by the implementation process at high level. Then there is the testing procedures, followed by the evaluation activity. Once this latter one has been accomplished the analysis process represents the final phase of overall system development life-cycle. In this context the sensitivity analysis can be particularly useful in case it can be used for the partial automation of the evaluation process.
In this case the example considered is represented by the application of sensitivity analysis to the dimensioning of the tank of the spacecraft which represent one of the recurring task that characterize the early development phases of space system. In particular the example considered involved also three different engineering domains as mission analysis, propulsion and structure. The implementation process is strictly related to the specific engineering problem that is considered. The decision depends on which parameters are available and which ones are not. These classification can change from problem to problem for the same set of parameters. The unknown variables must be computed while those available are used to define the boundary constraints for the case study. The objective of such approach is to clearly identify the sensitivity index of some input variables with respect to the output under evaluation, providing the information that the team leader can use to drive the study.

5.3.1 Jet Propulsion Laboratory - JPL

One of the first MBSE experience at JPL is represented by the Systems Engineering Advancement (SEA) initiative. This project was aimed to the identification of the potential improvements that the MBSE method-

ology can introduce for the management of space mission. The main efforts are addressed towards three different directions: the full life-cycle program (from early studies to operations and dismissal), the full depth within a project (from the systems down to components characteristics) and finally the full technical scope (considering all the various domain-specific environments such for example propulsion, avionics or electrical fields). The target of SEA project was to understand if possible improvements can be made for some of the functions directly involved in the space mission definition. This functionality regard mainly: system architecture, requirements management, interface definition, technical resource management, system design and analysis, system verification and validation, risk management, technical peer reviews, design process management and systems engineering task management. SEA activity developed product, services and training to accomplish this objective, focusing on processes, products, tools, people and technology.

SEA project investigated different modeling tools considering a set of criteria to assess their benefits in the management of real-scenario conditions. The evaluation criteria regarded the architecture and design modeling (considering for example SysML, UML languages or Enhanced Functional Flow Block Diagram EFFBD), the executable modeling and simulation (evaluating interoperability, trade space modeling and performances modeling for example), the information management (user-definable schema, metadata query, document linking, etc...), and finally the administration and usage.

As previously discussed the main purpose of SEA initiative allowed to better organize the knowledge about the competence model within the context of System Engineering. This perspective is characterized by the technical knowledge, the personal behaviours and the processes. Technical knowledge refers to the domain/discipline specific viewpoints that are involved in the system development process. SEA has also shown how one of the most challenging aspect is currently represented by the investigation and integration of model-based engineering design (MBED) tools.

Another interesting MBSE initiative has been developed at Jet Propulsion Laboratory in the context of space mission applications. Model based system engineering paradigm is mainly related to the work of Modeling Early Adopters group and Integrated Model Centric Engineering initiative. After an initial phase of feasibility analysis the main research topics regard the practicality and usability studies. The current model management capabilities are well increased thanks to a maturing standard and tooling interfaces that are addressed towards the applicability to actual space design programs. MBSE approach not necessarily entails a wide use of SysML language and its methodologies must not be confused with the conceptual architecture that SysML covers. Model-based architecture allow to manage multiple languages/tools and methods, defining at the same time different engineering perspectives of the same model. Analytical models has started to be integrated within this methodology, considering also the current workflow that characterize the development of complex systems. The configuration of multidisciplinary design environment can be obtained through a proper transformation from the SysML model to a multidisciplinary design environment (as for example ModelCenter, Phoenix Integration) [49]. The main aim of this approach is represented by an automated process for the generation of trade space for the support of analysis activities for components in use. The application of SysML language to products modelling has been addressed to verify consistency and completeness of model definition. Another interesting feature is directly related to the satisfaction of uniqueness for the involved elements (avoiding the potential presence of redundant data for the same object) and also to the definition of the necessary abstract classes to model what actually is needed.

MBSE methodologies can help to avoid design errors, supporting the project development in more consistent way. It becomes even more difficult to control the increasing number of design variables that can be identified during the development phases. A model based approach and an unified common tool for the management of system level characteristics allows to reduce the likelihood of wrong choices and design errors. Straightforward architectural design has historically lead towards errors that has caused the loss of space system. For example spacecrafts as DARTS or Mars Reconnaissance Orbit have been affected by errors that have negatively influenced the accomplishment of their mission. The errors that have compromised their mission might be identified by a more suited design process with a model based approach.

UML/SysML language has been investigated for the implementation of a model based design approach within the Jet Propulsion Laboratory experience. The formality and semantic rules that characterize the view diagrams allow to enhance the model consistency and parameter interdependencies. Currently the

research topic direction is addressed towards the definition of model management process where the system can be "compiled" from an evolving model. In this way it is possible to manage the right accomplishment of system performance with respect to customer requirements early in the design phases and also through a more formal workflow. The main features related to this aspect are related to the version control (for the current baseline of the system), tracing of the dependencies, data integrity, modularity and reusability. In this context becomes particularly interesting also the right management of changes propagation since clashing needs are often highlighted during the development phases.
Enhanced changes propagation allow to rapidly update system characteristics (evaluating different design solutions more quickly) but at the same time require a consistent check for the people that are working on the same system (that have to relate to system that change continuously on the other side). Another important topic is represented by the correct management of system alternatives investigation (for example how consider the possibility to manage different system alternative on different branches). Another important feature to be investigated is directly related to the capability to understand when a particular object can be considered ready for reuse, creating a library of well-defined and common accepted engineering objects (for example the version control of the released library becomes an important feature).
In this context SysML is an evolving standard and one of the main challenging problems is represented by the backward compatibility with previous version, since continuous improvement and changes are introduced. Project that last several years and the MBSE methodologies are applied from the beginning then an important effort must be allocated to a refactoring process in the case some new standard features have been introduced.
The last problems has lead the OMG to manage the standard versions reducing the time spent for the correction of previous introduced element that are not allowed in the new release for example. The main aim is represented by the possibility to reduce the risk related to the passing from one version of the language standard to the other. The transformation rules in this case must be well defined to avoid relevant inconsistencies between the models. The main direction of the research topic related to SysML development is currently represented by the analytical integration with external domain specific tools/solvers. Domain-specific tools can gain increasing popularity thanks to SysML interfacing and some classes of such instrument are reported in the following list.

- Requirement management tools (for example DOORS)
- Satellite Toolkit
- Math solvers
- Modelica tools (for example OpenModelica or Dymola)
- Mechanical CAD tools
- Electrical CAD tools
- Operational research tools
- Campaign simulation tools
- Process tools

The interconnection between the models in the domain specific environment with the SysML system model can currently be faced with different approaches since these an interesting research topic. Some examples of such approaches that face the problem of the interconnection between the domain-specific models (define in the various native domain environment) and central system model are reported in the following list (they are also currently the alternatives considered within the context of JPL research activity).

- Projection between models for interconnection in native domain (the properties coming from the domain specific environment are connected and loaded within SysML system model).

- Semantic coordination.
- Connect relevant parameters
- All models as views on same supermodel.

The most interesting challenge in MBSE is mainly related to the relationship between the computer science methods (considering the SysML development environment) and object-oriented approach to manage the current engineering design process.
The benefits coming from MBSE approach have been recognized from the JPL experience and can be expressed in the following lines. Coordination and enhanced traceability of the product components are some of the main advantages. Another interesting feature is related to the possibility of automated verification and also generation of documents (directly related to the reduction of time and costs). The formal definition of rules and connection between the models allow to support the generation of cases and operational scenario. Finally the definition of a unique central model can enhance the capability to query the needed information.
Another example related to the application of MBSE methodologies to space mission concept can be identified in the Europa Mission Concept Study [50]. Jet Propulsion Laboratory has gained important advantages from the use of MBSE methodology within the design of space mission. In particular the Europa Mission Concept Study has been done through the integration of MBSE approach which allowed to capture and analyse the system solutions more effectively. This study has enhanced the importance of system modeling for the management of space systems architecture. This approach allows to better manage the complexity of the system considered, providing the capability to manage the dynamic architecture solutions that typically characterize the early development phases. This feature shows a better behaviour in the system design process with respect to the traditional system engineering modeling.
The reference case used to assess the MBSE methodologies is represented by the Jupiter Europa Orbiter (JEO). The development of this science mission in the design Phase A was supported also with the partnership of the IMCE initiative. In this case SysML has been chosen as the modeling language for the integration of MBSE definition architecture. This environment allowed to integrate in the same framework all the involved stakeholders, providing a common tool to share information and discuss about the possible solutions in more consistent way.
The MBSE approach allowed to analyse consistently different mission configurations, exploring other possible solutions and proposing the split of the original architecture into two independent solutions. In this case the science instruments are properly placed on the two independent spacecraft since the configurations considered show improved performances with respect to the original concept. The architecting information must be managed in a more comprehensive system model as the design becomes more detailed. A modest architecting framework was developed to accomplish this objective and it was subsequently adapted using an open-source web development tool for collaborative databases. The tool developed within this context is referred to as Architecture Framework Tool (AFT).
Further improvements can be obtained with such tool types in order to support the architecting effort. This approach will allow to better manage the workflow related to the design process, encouraging a deeper iterative and incremental approach in the development of the product.
The collaborative SysML tool environment chosen for this study has been identified with the commercial solution proposed by NoMagic and represented by the MagicDraw tool. The deployment of such tool has been characterized by a well-supported training phase for the people involved in the system modeling process. The SysML environment has been properly adapted to the specific needs of the modeling team. A modeling plan has been developed to drive the system evaluation with a more flexible approach within the MBSE paradigm. In particular the mission conceptual architecture description is implemented within the ATF framework (in particular three mission concepts are evaluated). The single mission concept is then modelled with SysML defining the physical decomposition, system and subsystem block diagrams and mass reports. In this way the team iteratively interact with the system model and the contained information, exploiting also the capability to partially automate the generation of documents or some preliminary analysis (such as the mass budget).

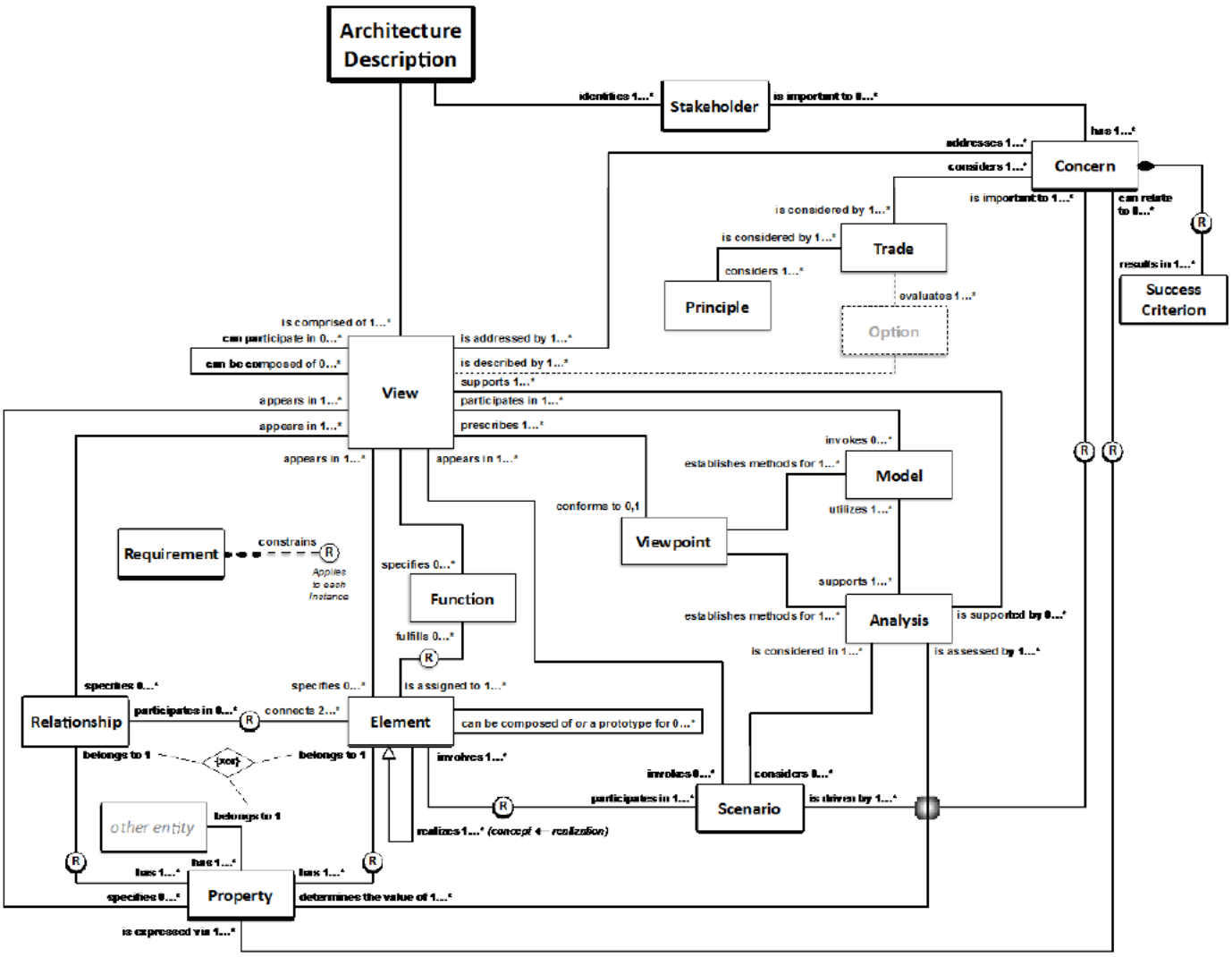

Figure 5.2: Conceptual meta-model of JPL research initiative on MBSE [50].

The description of the architecture is defined on the basis of the concepts inspired by [51] and developed to properly manage the representation of the various features that characterize the system model such as hardware, software and operational aspects. The result of this conceptual analysis is reported in figure 5.2.

The block elements of the previous figure refer to the categories that conceptually describe the space mission architecture. The various categories defined in this diagram define a group of information that the related models must contain. In particular the system components must modelled following the relationships defined with this diagram. A similar approach has been considered for the generation of the system meta-model defined within this work but with slightly different categories and their relationships. The architecture description follows the relationships and conceptual categories defined within the previously introduced meta-model and all the contained information have to been integrated within the OWL ontology to consider the possible interface with web-based services. The following considerations briefly characterize the objects involved within the definition of the space mission architecture.

The Architecture Description identify one or more Stakeholder, each with one or more Concern. The Concern itself results in one or more Success Criteria (this object is directly related to the integration of the requirements coming from the stakeholders). The Concern is considered by one or more Trade and each of which considers one or more Principle. The Trade element evaluates one or more Option. The View element is directly related to the Function, Element, Viewpoint, Relationship, Property, Concern, Scenario and Model categories. All the interfaces that potentially characterize the Element definition are all modelled through the Relationship object. In particular this categories include also all the possible elements that are associated to the interaction between the Elements that compose the system. Model category is

used to support the analysis process which is related also to the definition of the information contained within the Scenario element.
The Trade element has been introduced to group all the possible Option that are referred to a certain Concern of the Stakeholder object. Certain Option is described by one or more View and within this element the description is obtained through the definition of the Element and other objects.
The system definition starts from the proposed model and then consider also the investigation about the hierarchical composition of product elements. In particular the components of certain elements can be integrated in different manner obtaining various configurations that can be studied to assess their advantages with respect to another one. The process of integration of atomic elements into a composite product is also identified as deployment of such product. Such hierarchical analysis has been captured through the Internal Block Diagram of SysML language. This diagrams allow the development team to discuss about the interconnections that are modelled for example.
Another important feature that must be considered is represented by the work breakdown hierarchy. This aspect is mainly related to the organization of the work package and its decomposition, allocating the various resources on the system elements. The management of space mission concept through a level decomposition based on the distinction between the equipment and subsystem often results in an oversimplification. The Viewpoint category has be introduced mainly for this reason while the system hierarchical definition is still maintained to ensure a well-defined organization and modularity of the analysed system.
Analysis process is as important as its correct documentation. The management of the technical margins (as for example those related to the mass, the power or also the energy) is one of the critical issue that characterizes system development also in the early phases. The evaluation of such technical features can be done with two approaches. The investigation can be realized within the SysML tool itself if the analyses are not so computationally demanding. The computation can also be interfaced with an external solvers when the solution is particularly demanding.
Another interesting feature that can be found among all the possible capabilities is represented by the support for the generation of the Mass Equipment List (MEL) which is directly related to assessment of mass budget assessment. Starting from the root node of the product the mass budget can be obtained iteratively processing all the contained items.
The same modeling approach can be used to monitor and manage the power margin and energy balance to evaluate preliminary analyses. The SysML implemented model does not still consider the influence of time and so the evaluation of power budget is referred to static preliminary computations. This approach is however not well suited to more detailed design since the time scheduled components operational modes and scenario directly affect such investigation.
Data balance margin can be evaluated and managed similarly to the previous section since the notion of time is not still well implemented within SysML environment.
The other features evaluated within this study are represented by the radiated equipment lifetime, science margin, cost estimation, integration with cost models and finally the automated report generation and web publishing.
The radiated equipment lifetime and margin (RELM) model has been developed to assess the effectiveness of the current components shielding or if it is required a better protection. The computations of such evaluations are demanded to an external solver (Wolfram Mathematica) for the processing. The science margin helps to identify the efforts required to address a science concern. A Science Margin Model (SMM) is used to quantify the balance between the changes in technical design and the corresponding changes in science return.
The cost estimation is one of the critical element for a quick evaluation of solution feasibility and also for the right identification of the resources needed for the project. Most cost models are related to the mass parameter in the early phase of the development and since this variable can be estimated more consistently and before with respect to the traditional approach.
The report automatic generation is one of the most interesting benefits related to the application of an MBSE approach to a space mission project. Reports, tables and documents can be generated on the basis of the information available in the system model (from the diagrams for example), allowing a better control and consistency of the data introduced.

Some other example of such integration can be found in technical literature as the work [52] where MBSE methodologies are applied for the analysis of space mission operational scenarios. In this case the Radio Aurora Explorer (RAX) mission is modelled using a SysML tool within the context of MBSE methodology. The choice of a cube sat mission allows to show the capabilities of such a modeling paradigm in the context of highly integrated and coupled subsystems. The closeness of the involved subsystems and the high level of equipments integration lead to a particularly challenging design process for such a missions. In this case the MBSE methodology has been considered for the management of the behavioural and operational aspects. Several simulation tools has been integrated to assess some analysis on the basis of the information contained within the SysML model. Data included within the behaviour models, subsystem functions and internal states are used to set up the simulation scenarios for the spacecraft. The main aim has regarded the demonstration of the applicability of such approach, investigating the feasibility, the evaluation of performances and the computation of system metrics. The information contained in the central model has been used to build the representative mission scenarios simulation, highlighting also the feasibility of operational schedules evaluation. This modeling architecture enhances the capability to obtain operational performance feedback still in the previous design phase, allowing to proper identify potential development errors and also reducing the problems related to the consistency of the data exchanged.
Different analysis approaches has been considered since various strategies can be implemented for the evaluation of the required parameters. In this case an example of such analysis has been realized exploiting the internal solver available with the SysML modeling tool. Such capability is based on the evaluation of the design parameters that are defined within a parametric diagram. The relationships between these parameters model link between the quantities that characterize some physical law or mathematical expression. The modeling approach of SysML parametric diagram allow to define the contained elements without defining which of these ones are outputs or inputs. This acausal representation of the relationships between some parameter allow to consider the same rule/law also in the case of other evaluations, when for example the quantity that have to be computed is now an input when in another context this one was an output. In particular the internal solver used in this work is called ParaMagic and it is available within the SysML MagicDraw modeling environment. The parameter/s to be evaluated are identified on-the-fly by ParaMagic and the causalities between the available data are assigned consistently with those available. This analysis instrument allow to realize some preliminary evaluations without the need to link to an external solver for the characterization of some scenarios. This method is well suited in the case of simple relationships between the considered parameters but the implementation of an high number of values can be cumbersome and not particularly easy to understand. Also this analysis instrument is not suited in the case of models slightly more complex or when the parameters to be set are many since the substitution for the single scenario must be done manually. In this context it is necessary to instantiated the investigated scenario starting from the block definition diagram of the analysis and system design elements. In this work the ParaMagic solver approach has been used to solve the communication download analysis.
The power analysis is instead performed using the PHX ModelCenter tool to model the workflow between different external solvers. In particular external solver are used to compute the orbital position of the spacecraft while Matlab scripts allow to solve for the dynamics of the satellite. Finally the mission system activity analysis has been performed using the Cameo Simulation Toolkit (a plug-in of the MagicDraw tool), animating the state machine and activity models. In this way a behavioural analysis of the model has been performed, checking the data, information and logical flow between the various elements involved. One of the major effort that has been enhanced from this work is identified with the time spent for the integration of simulation object and their testing since they are integrated in a common environment. This process require the definition of different files and scripts that allow to manage the processing of the information from one solver to the other (generating the related wrapper functionality). Further improvements are scheduled in this direction to provide a more consistent simulation environment.
Interesting results coming from the integration between different types of data and geometrical models are also available from [119]. Such work basically concerns the capabilities that can be achieved through the use of Building Information Models (BIM) for the management of the data of a complex project. The connection of various types of analyses beyond pure graphical representations allows to improve the ef-

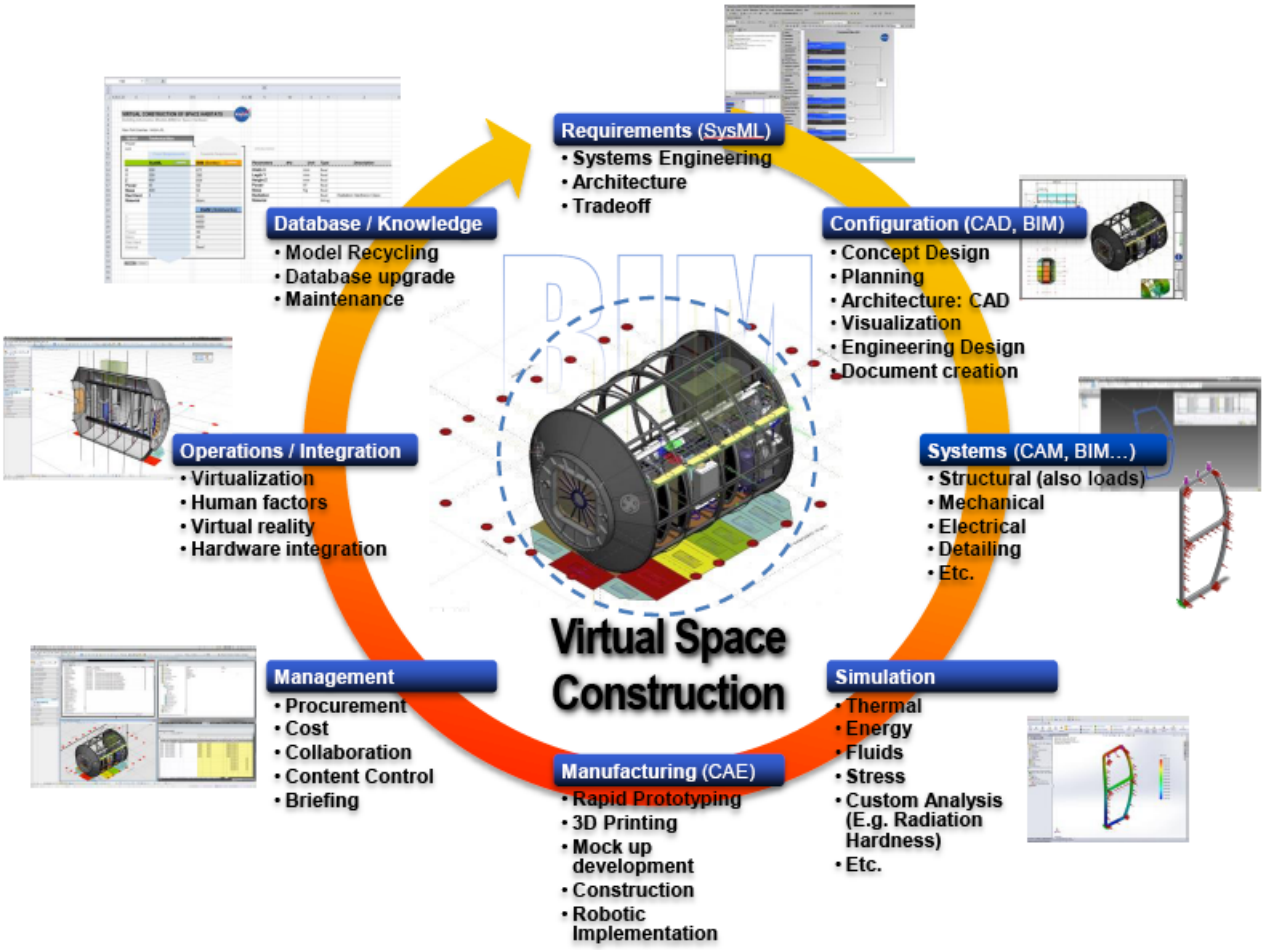

Figure 5.3: Conceptual overview of the lifecycle of an aerospace system and the phases that can be covered with the proposed Virtual Space Construction Process (VSC) [119].

fectiveness of data exchange. In particular it is investigated the integration between model-based systems engineering languages/processes (for example SysML) and a powerful geometrical architectural design tool with BIM capabilities. The related approach is applied on a hypothetical example concerning space habitats in order to evaluate how and in which way the design of complex system can be enhanced in the near future. A conceptual overview of the workflow of aerospace hardware development and the phases that can be covered with the proposed Virtual Space Construction Process (VSC) is provided in figure 5.3.

5.3.2 TU Delft

Another interesting initiative related to the integration of Multidisciplinary Design Optimization and Concurrent Engineering has been represented by the research activities that are developed at Delft University of Technology [53]. In particular the academic effort has been addressed towards different directions as the operative research, education and application. The developed methodologies are evaluated in the context of actual space applications as cube sat projects (for example Delfi-C3 and Delfi-n3Xt). At the same time the model based system engineering methodology is currently proposed in different university courses while also multidisciplinary design optimization research activities are analysed in the same context.
The application of concurrent engineering for space applications has enhanced certain limitations as this research group has observed from the actual design approaches. In particular the main limitations can be summarized in the following ones (also reported in figure 5.4):

- No multiple options
- No trade-off

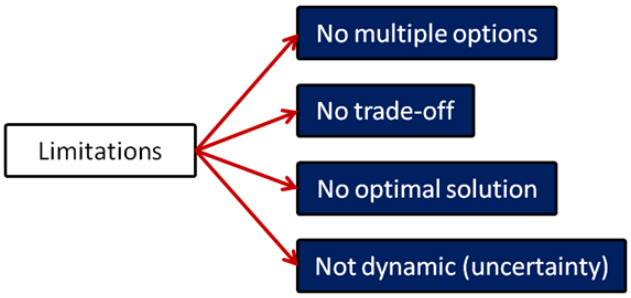

Figure 5.4: Overview of the main limitations of the concurrent engineering for space.

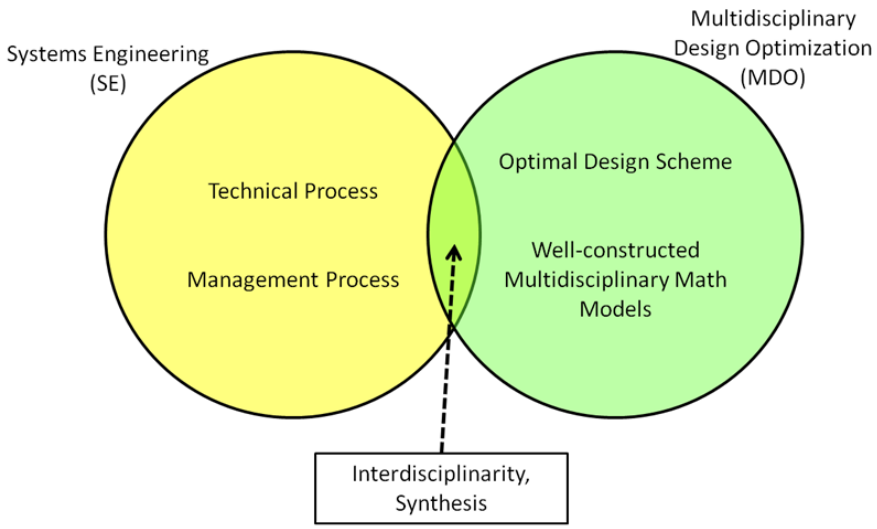

Figure 5.5: Main features and common aspects of MDO and System Engineering.

- No optimal solution
- Not dynamic (uncertainty)

All these elements come out in the context of a multidisciplinary team working on the same project. The same problems can be remarked for certain collaborative environments (as for example within structures similar to Concurrent Design Facility). In particular the interaction between people working on the same product at system level shows as the management of multiple options is difficult to formalize. The same conditions lead to a not well established definition for the generation of trade-off and consequently also for the identification of optimal solution. This situation is then characterized often by a not dynamic environment where is not so easy to manage the uncertainties that can be met during the design process. The System Engineering area is well described by the definition of technical process and management process that in the last several years has been relatively formalized (as can be seen from the NASA System Engineering Handbook). The Multidisciplinary Design Optimization environment is mainly characterized by the definition and a clear understanding of the optimal design scheme. Another important feature is represented also by the construction of a well reliable multidisciplinary math models. The contact points between such areas are represented by the synthesis and the interdisciplinary (figure 5.5).

One of the most interesting objective of this study is represented by the assessment of the feasibility related to the integration of MDO into existing System Engineering/Concurrent Engineering architecture. One of the key feature that characterizes such potential integration is represented by a clear understanding of the overall interface and synergies that can be identified when different design processes are put together. From this perspective the MDO approach used covers a key-role in the definition of the product development. The interrelations between different models and various design activities figures out how difficult is to put all together the integration between the MDO techniques and SE/CE methodologies.

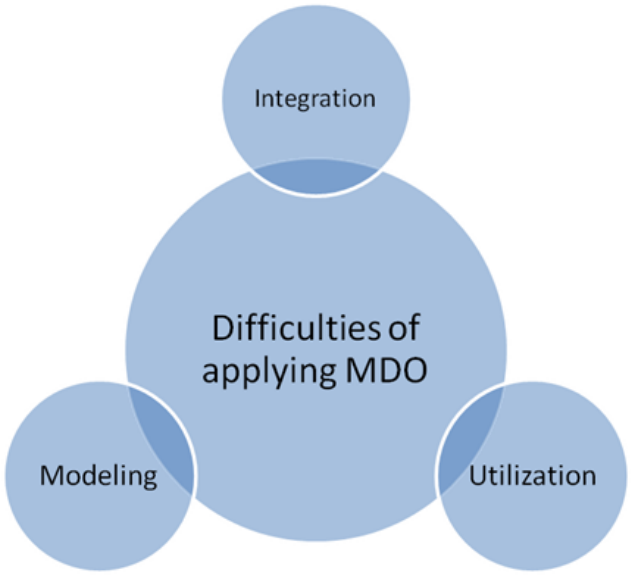

Figure 5.6: Main areas directly involved in the integration process of MDO techniques.

Some of the difficulties that characterize the application of MDO can be resumed in the following areas, represented in figure 5.6 for the sake of clarity:

- Integration
- Utilization
- Modeling

The incorporation of MDO paradigm within SE architecture requires a clear definition of different involved elements. The main idea is represented by the incorporation of automatic search of optimal solution under uncertainties and the information available in quantitative, qualitative and uncertain form. The other objects to be considered are the knowledge and the existing S/C SE framework.
The analysed methodology is based on the principles of Knowledge Based Engineering (KBE) where all the elements that characterize the definition of specific system are formalized following a certain pattern. Starting from requirements the space system is then decomposed defining the space segment (bus and payload) and ground segment, proceeding through all the levels to clearly identify the required features. The same project has considered the evaluation of such integration on a case study represented by a distributed space system. This scenario has also been modelled considering also the identification of uncertainty source, providing the base for a problem of Uncertainty Multidisciplinary Design Optimization. Independent input (design variables) have been managed to proper evaluate the dependent output (attribute values), obtaining the optimal configuration for the conceived optimization problem.
The overall system model has been implemented considering also the definition of RAM model, lifecycle model and cost model. In this manner the traditional technical models are kept separated from those that are more related to the management perspective of a product. For example the mission analysis model, spacecraft model, launcher model and ground segment model are all included in what is called a performance-based context (also if the previously defined group of models can also be considered for the evaluation of system performances). The RAM model is used in particular to evaluate the reliability, the availability and the maintainability on the basis of lower level system (contained elements) reliability, TRLs, redundancy, etc. The lifecycle model is instead used to define the list of activities that are needed to proceed from user requirements to the a specific phase. The cost model is based mainly on the estimation of three principal sources. In particular these are represented by the development costs, the launch costs and finally the operational costs. This approach for the evaluation of the costs related to the investigation of overall system costs can be defined in different manner on the basis of different methods and of available information. The current challenges for the integration of MDO techniques are conceptually reported

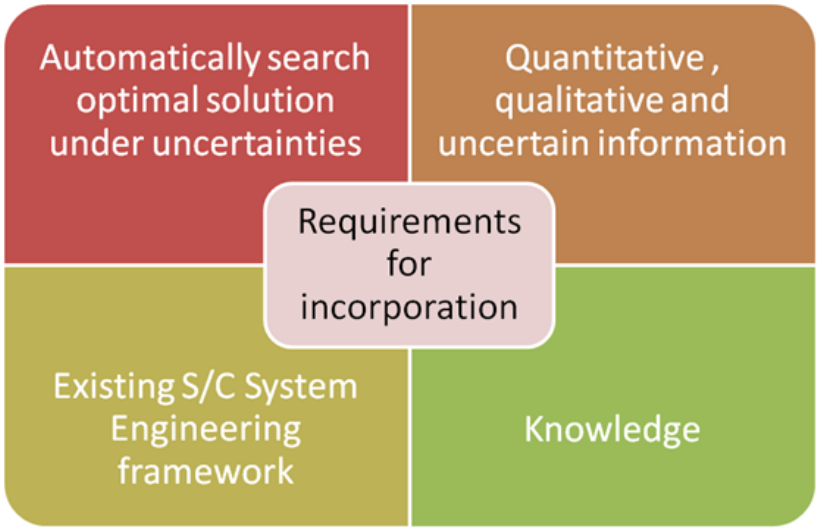

Figure 5.7: Overview of the main challenges for the integration between MBSE environments and MDO capabilities.

in figure 5.7 from the perspective of TU Delft analysis and experience (they are referred to the present practise for concurrent engineering for space).

One of the most important phase is represented by a clear understanding of the interconnections between the various models and in particular the feedback and forward links that can strictly couple two or more simulations.
The problem has been modelled through the assignment of uncertainties to system model parameters and the incorporation of uncertain events. In particular the lifecycle model includes the possibility to stochastically introduce delays within an activity. In the same manner the component failure can be modelled as a distribution. Finally the CER model considers the effects of probability distribution to properly evaluate the uncertainties that can affect the costs estimation.
The integration of MDO into SE/CE framework has been conceived through the definition of concepts trade space, solutions identification and assessment, design parameters investigation and finally system/subsystem evaluation. The final aim of such approach can be recognized as the identification of the optimal design solution.
This research activity is focused on the assessment of such integration and future developments are addressed to the interfacing with the Concurrent Design Facility available in TU Delft.

5.3.3 University of Michigan

The integration of MDO techniques with an MBSE environment is currently investigated with the support of different research initiatives, as can be seen for example from the works developed in the context of University of Michigan. In particular some interesting research activities, as reported in [69], [52] and [93], show how the integration of optimization techniques within a MBSE framework is a promising approach for the development of complex aerospace systems. In this case the development process of small satellite systems has been supported through the use of a SysML tool integrated with external solvers. SysML has been used to model all the representative information of the system itself, allowing also the definition of the rules and laws that characterize the relationships among the properties of the satellite. In this case the parameter diagrams have been used to link all the values that are directly related with each other for the computation of a certain variable. The SNR Analysis link budget for example has been built with such an approach. All the data required to define the current status of the project are stored though SysML, used also to define the topological architecture of the subsystems modeled for such reference case, considering however the preliminary phases of a project. The attitude determination and control subsystem is modeled with all the related components in the same environments that allows also a clear and con-

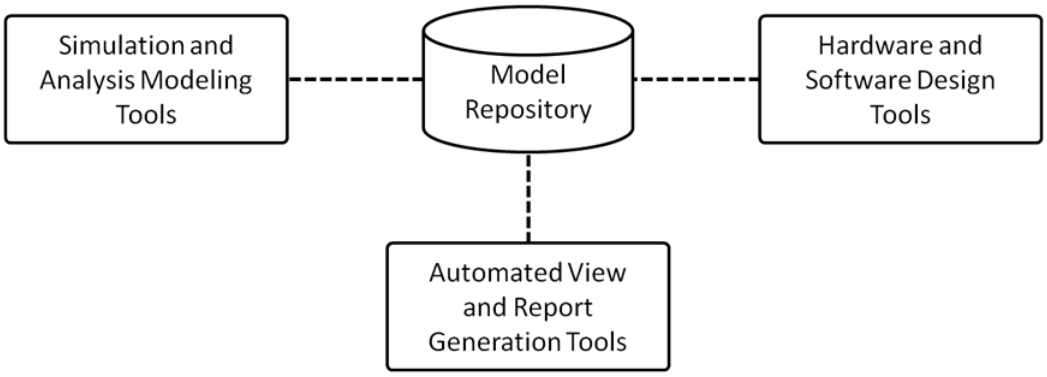

Figure 5.8: High level representation of the infrastructure considered for the design problem of CubSat example.

sistent representation of all the involved elements. The SysML system model for the Cubsat mission is then connected with external solving environments through proper developed interfaces and plugins to compute the needed quantities. In this case the simulation capabilities mainly regarded power analysis, communication download analysis, mission system activity analysis and orbit dynamics. The information collected within the system model are also used to set up optimization analysis through the use of developed interface with a multidisciplinary analysis tool (PHX ModelCenter®in particular). The information are properly processed to partially automate the definition process for an optimization analysis, directly managed within the ModelCenter environment with the data provided by the System Model. Such environment monitors the overall execution of the optimization as well as all the simulations required to achieve the desired results (for example Matlab©codes and Excel ©spreadsheets are also included within the cycle). In this case the integration between MBSE environment (represented by the SysML System Model modeled with a commercial tool) and the MDO techniques (provided within the context of PHX ModelCenter®) is obtained though proper developed interface plugins and script that manage the generation of the overall scenario. An high level representation of the model based approach can be conceptually represented in figure 5.8.

The main advantages highlighted by such approach are represented by the capability to partially automate the generation of simulation scenario on the basis of the available information. In this way it is possible to reduce the consistency problems that can arise when model transformations or data exchanges are required to built a simulation case. At the same time has been possible to better exploit the overall information to define an optimization study that was able to better explore the design space with less efforts from the user. At the same time some challenging issues must also be properly faced.
Different licenses are often required to properly set up the used simulation tools, considering also the related vendor support in some cases. This problem often rises with the use of commercial tools and their related environments while the alternative solution is represented by open-source software. In this case on the other side the documentation and support is often not necessarily ensured by the developing teams. Another problem highlighted by such studies is represented by the large efforts required to set up the simulation environments, including creating wrapper files, wrapping models, saving and re-opening models.

Chapter 6

Conceptual Infrastructure

One of the most important features directly related to the developed framework is represented by the need to analyze the potential introduction and formalization of trade-off capabilities within a consistent modeling environment. The evaluation of system performances and the study of feasible solutions/configurations represent some of the most important activities of a system project. The early development phases are mainly involved in the definition of product characteristics and heavily affect the following system behaviour. This process is lead by the correct identification of the criteria that are then used to investigate the responses to external environment. A formal definition of the criteria used to quantify the effectiveness of certain system solutions is a complex activity and often is strictly related to personal knowhow of single person, driving sometimes to a subjective perspective. The focus of the current work is also represented by a feasibility assessment about the possibility to formalize the overall architecture evaluations. The proposed approach theoretically shows interesting benefits regarding the reduction of possible misunderstandings and subjective interpretations of system results.
A correct evaluation of an architecture has generally to take into account different elements:

- Requirements
- Scenarios (operational and not-operational)
- Stakeholder's concerns and related preferences
- Overall architecture properties

All these elements must be used to properly support the decision making process, providing the instruments to justify certain choices and correctly investigate impacts on design solutions. Such aspects can be differently approached and managed by various system modeling tools. All the previous general considerations can in fact be processed developing different high level formalization often on the basis of the company or organization know-how about System Engineering. The key-role in such procedures is also represented basically by a correct understanding and conceptual definition of the metrics and evaluation criteria for the investigation of the response under analysis.

6.1 Introduction

The previous sections have highlighted the current needs with respect to the research field of System Engineering. In particular the considered examples and research initiatives had the role to show how a formalized infrastructure for the management of the advanced phases of a project is currently not properly defined. The definition of a consistent conceptual architecture that includes the great part of the available scenarios is difficult to obtain, above all for the most advanced phases of a space product where the possible cases cover a wide range and a unique set of common concepts is difficult to establish. The main aim of the present work focuses on the understanding of the possible alternatives that can be chosen and then the development and investigation of one solution among these ones. In this way the final purpose

is addressed towards the identification of the missing elements and concepts that must be taken into account. The application of model-based methodologies in the advanced phases of complex system requires in fact a well-understanding of all the involved entities (from people to the processes and design methods), avoiding the possibility to neglect some modeling methods or analysis mechanisms. This aspect must be not undervalued since it strictly affects the evolution and effectiveness of the methodologies associated to the Model Based System Engineering "philosophy".
The next sections first provide a description of the common issues that that can be found during the application of a model based methodology in the advanced phases of a development process. Then the main definitions of the terms used are reported. The classification of the involved concepts as well as the relations among them is reported, presenting what can be considered as a simple taxonomical or ontological analysis that paves the way for the overall infrastructure. The main concepts are presented in this section but more details are provided in the appendices and they are used to conceptually build the framework that can then be implemented following different alternatives (for example the same conceptual infrastructure can be implemented differently on the basis of the chosen technologies).

6.1.1 Current issues

Product lifecycle management (PLM) is an all-encompassing approach for innovation and system information management from preliminary concepts to end of life. PLM shows its capabilities to support model based systems engineering and system lifecycle. Information models, meta-data models (different viewpoint to represent product model) and procedures need to be developed in order to support multi-domain systems engineering, simulation-based engineering, and knowledge management, besides the current design approach. The difficulties that can be encountered during data-exchange are partly caused by lack of general accepted industry standards and protocols for PLM meta-data models and processes. Information models and data exchange between virtual prototyping solutions and PLM systems need improvement. Additionally, virtual environments are getting larger and more closely integrated together often through proper developed network system. In this context a more effective data structure and collaborative methodology cover a key-role for the definition of the right solution.
The use of a collaborative environment has emerged as a consequence of a continually changing working place which calls for the collaboration of multiple actors, with different background, roles, knowledge, expertise and tasks. The capability to collaborate over time and space, within and across organizations or corporation, is fundamental to reach this flexibility through the best possible management of the knowledge and resources available (ensuring for example a well-defined way of information access). The desired flexibility can be pursued through the implementation of a distributed environment with particular emphasis on interaction mechanisms among all the involved actors.
The current implementation of system modeling infrastructures for the management of overall information, such for example SysML solution, has shown interesting features and numerous advantages. SysML models pave the way to the definition of a well-harmonized system representation that can widely reduce the development time and costs with respect to the traditional design approaches. Such capabilities are highlighted by the various reference cases and studies that are carried out on MBSE methodologies through the use of SysML language. These analyses have also enhanced the difficulties that come out when a static and representative model is interfaced with simulation environments. They show not only the benefits that can be obtained by such an approach but concurrently also the main fields that require a better definition. The latter ones slow down in fact the application of such methodology up also to simulation and analysis environments. Some of the points already opened with respect to SysML language are reported in the following lines. One of the main aspect refers to the practical usability of this language for system knowledge and the related easiness to be learned, used and employed in practise. Such languages is also evaluated to understand its applicability in daily problems with low overheads, considering for example the capability not only to manage a restricted subset of problems.
Another important criteria that must be take into account in the choice of SysML as system modeling language is affected by the degree of independence. The capability to run across heterogeneous environments independently by the related platform is a key-factor for the definition of a collaborative environment. Such

issue must be properly evaluated with respect to SysML tools and platforms to figure out if integration problems can arise when different organization and corporation work together (for example in the case some resources are shared in the same project).
The investigation of SysML language involves also its capability to provide a zoom function, enabling in particular the users to start at a high modeling level (for example at overall system level) and then navigate to mode detailed levels (as for example the component or production line level). Various research initiatives are currently addressed to the evaluation of such kind of feature since zoom capability can widely improve the actual design process with respect to the traditional one. In this way it will be possible to use the same system modeling environment both during the preliminary phases as also in the more detailed ones. Analysis activities are now considering the application of such capability to more complex project with the final aim to identify all the related issues.
Another important ability that languages as SysML must provide is represented by the effective management of system knowledge across different modeling domains. This capability is strictly related to the future developments of increasingly collaborative environments and must be well-demonstrated. The scalability of modeling languages must also be considered since the management of different types of projects running on different types of companies is a key-factor for the development of a shared modeling approach (involvement of Small Medium Enterprise – SME and large companies). The ability to readily react to changing conditions (as design changes or organization changes for example) is currently one of the features that is pursued and it is important that such agility is available from modeling languages. The actual demanding design environment is searching also for modeling solutions that are not necessarily bounded to proprietary aspects since in this way it is possible to develop their own environment with no limitations on licenses, customizing the framework on specific needs (no costs are associated to the licenses management and upgrading). The other benefits is also represented by the reduced or even absent installation and upgrading costs.
All the efforts involved in such research activities are mainly addressed to the evaluation of the actual economical benefits that modeling languages such as SysML can effectively add to product development and manufacturing processes. The main interest is addressed to understand how much the results of such modeling languages outweigh the costs of their adoption.
The use of an infrastructure based on SysML language have to face often with all the problems related to its application within a collaborative environment. The advanced phases of a project pursue in fact a wide collaboration among the involved teams and actors for the successful achievement of the objectives set. Currently the integration of a SysML tool wit+hin a collaborative environment is not well defined since all the available solutions are based on desktop applications without common procedures for the exchange of the involved information. In this case the collaboration among different people become difficult to achieve since the collaboration among person working on the same project requires proper developed merging mechanisms. Project data stored in the same file are difficult to manage when a large number of people work on it. Such process needs the correct handling of updates, accesses and ownerships of the edited information to avoid data losses and consistency problems. Currently the large part of SysML based tools provides functionalities that are particularly useful and well suited for the management of a large set of data on the same file. The management of data across different users working not on the same file (but for example on copies of it) requires mechanisms and capabilities that can be not necessarily embedded within a SysML based platform. In this case alternative approaches can mitigate such problems through the use of different types of infrastructure. In particular web-based technologies can improve the collaboration process and the current proposed infrastructure has been conceived to enhance such aspect with respect to the response performances that can be achieved trough a desktop application.
The primary objective of this work regards the analysis of a possible alternative solution in the choice of system modeling language. An evaluation of modeling approach is conceptually defined, formalized and evaluated starting from the just described considerations about system modeling languages main issues and features. In particular the application of an MBSE methodology is investigated considering the design of space systems in the advanced stages of the project. One of the current most challenging topic concerns the integration of such recent philosophy with the analysis environments. The capability to interface a modeling environment with analysis and simulation ones covers a key-role for the spreading of

model-based approaches. The proposed framework has been evaluated mainly on the capability to manage simulation models through a web-based interface, also developing a process for the definition of multidisciplinary analyses. The focus has been represented by the development and assessment of a framework for the definition of multidisciplinary surveys as soon as possible, theoretically conceived to support system performances analyses during both the preliminary phases and the more detailed ones. The developed framework is based on a different design approach with respect to the current solutions. Different research initiatives have basically driven the definition of various desktop applications that allow the definition and analysis of aerospace systems ([54], [55]). Open-source projects and proprietary initiatives have implemented different kinds of design and dimensioning toolkits, each one concerning a specific problem as radiation transport and effects, micro-meteoroids and space debris, planetary environments, contamination and spacecraft plasma interaction for example. They are generally not so flexible when the design scenario moves away from the nominal one since they are often hard coded around a specific design issue. The customization of such tools becomes not so easily to handle even supposing the possibility to modify some application parts.
The final purpose of the present work is not aimed to the formalization and implementation of such a design infrastructure since the commercial solutions and built-in-house toolkits offer already a wide range of performing capabilities. There is no need in fact to reinvent the wheel since the main issues come out when such analysis environments are integrated each other in a more complex framework.
The primary objectives are the development and assessment of a design approach that enable the integration at high level with no limitations on the analysis workflow that is often strictly related to both company knowledge and project needs.
The definition of a different approach for the design process can enhance the capability to properly exploit the available resources, customizing the simulation utilities and toolkit functionalities. The main idea is to develop a problem solving environment where the users can access analysis tools but are not limited by their native implementation. A multidisciplinary analysis environment where the design flow is not constrained by the built-in implementation of the framework shows numerous advantages. In this way the design workflow can be supported and monitored across its phases through the same environment that can be used in the same manner on other projects. Such feature enables also the possibility to seamlessly exchange data between different projects since all the objects follow the same data structure and are implemented on the same platform. In the end such modular approach improves the reuse of already defined components both from different project developed at the same time as also from previous ones, exploiting the available historical data. The concepts, main features and related results are presented with more details in the following sections.
The right representation of system configurations is difficult to achieve and different solutions can be considered. Such choice directly affects the development phases of a complex system and different research activities are addressed towards such aspect as [111] for example.

6.2 Taxonomy

The definitions considered in the current work are based on the concepts available from ECSS technical memoranda and are slightly modified in some specific cases [56]. Such changes and additional integrations to already developed definitions are included to take into account aspects that are not initially foreseen or are not covering some particular situations for the proposed methodology.

- **Actor**: In the current work the term actor identifies the entity that acts through specific means with other entities. In particular such integration can involve both human users and computer systems, depending on the specific situation and involved entities. For example during system integration and manufacturing there are potentially human actors that interact with a product and its components. On the other side a web service interaction can involve a human user and a client machine that represents a lifeless entity which however can be implemented to provide all the responses needed

to interface with other actors. The identification of actors and their relationships within a specific infrastructure depends on the context and level of details that are considered.

- **Analysis**: Analysis represents a verification method that uses techniques and tools to confirm that verification requirements have been satisfied. Basically an analysis can be done through different means. A specific analysis can be done with the support of simulation models and tools but in other cases analyses can be realized without such elements. For example a mass budget can be considered as an analysis activity but it not necessarily involves a simulation (such affirmation must be properly understood with reference to the concept of simulation). Generally it is also possible to associate analysis with an analysis model that can be used for example to set up a specific simulation. In this manner the same analysis model can in fact be used to generate and manage different simulations. The relationship between analysis and simulation classes places the related concepts on the same level. The analysis concept can be defined independently to the fact that it is associated to a simulation items. In the same way a simulation can be used not necessarily within the context of an analysis since this method has been conceived with the final aim to verify one or more requirements. A simulation is conceived instead to model and foresees the behavior of a product before its actual realization or also to evaluate the possible responses of a particular object (also if already manufactured) in some particular scenario before certain actions are taken. In this case the simulation is not done to verify some requirement (providing support to an analysis activity) but only to foresee system behavior. Both concepts are quite similar but their main difference is related to the final purpose they are associated with. A simulation item can be then defined independently with respect to a particular analysis. A simulation can be linked to a specific analysis when it is done with the final aim to assess the product behavior with respect to a certain requirement. Generally the individual analysis can be associated to a number of different simulations since the verification of a particular analysis situation can require the execution of different kind of simulations. In particular in the preliminary phases of a project the single analysis can be supported with a number of simulations. For this reason the two concepts are considered separated to avoid possible misunderstandings.
  Some considerations can help to further describe such concept but in the following expressions more concepts will be clarified. Considering the common terms used in System Engineering an *Analysis* can be linked to an *Use case* while the the Simulation (conceptually related to the concept of *Simulation case*) can be linked more properly to a *Scenario*.

- **Baseline**: A baseline represents a set of information which describes exhaustively a situation at a given instant of time or over a given time interval. A baseline is generally used as a reference for comparison with and analysis of subsequent evolutions of the information. In such definition systems options and alternatives can be considered or not as object belonging to the current baseline on the basis of the desired modeling purposes. In the current work the key point for baseline definition depends on the choice for options and alternatives belonging. The term baseline should not contain options and alternatives if literally considered on the basis of ECSS definition. In the same way the design variables provided by the users must also be managed as external object to the current baseline but must be however traced within the same project.
  In this sense the options and alternatives can be linked to the related project and a specific baseline at the same time. They are directly contained within the project but not in the baseline, ensuring that when the baseline is deleted the linked optional or alternative objects are not removed from the project. Baseline is basically used to take account of the nominal representation of the overall system as an instantaneous shot of project nominal state (current configuration without options or alternatives) in a specific time instant.

- **Dataflow**: An important concept that must be clearly defined and that is strictly related to the integration of multiple analyses within the context of a multidisciplinary environment is represented by the dataflow. This term must be not confused with the concept of workflow that is however defined in the following. The dataflow describe the flow of information that characterizes the execution of a

certain pattern of analyses. In particular the dataflow basically describes the relationships between analysis objects within a specific scenario with respect of input/output connections. This definition does not include show the time relationships between the involved elements but shows only the dependence between the variables. The same dataflow can in fact be managed in different ways on the basis of the time scheduling chosen for a particular case. The time dependence is not highlighted with the dataflow but with the workflow.

- **Design**: A design can be seen as a set of information that defines the characteristics of the product. This definition can be partially related to baseline term since their meaning is quite similar. In the current work the term design also includes all the options and alternatives entities that come out during the project development. In particular the design refers to the options and alternatives that are currently under investigation and not necessarily to only the nominal configuration. From this view point the concept of design is wider than baseline one which in this work refers only to the nominal set of system data.
  Design can equivalently be considered as the process used to generate the set of information defining the characteristics of a product. In this case it refers mainly to the design activity than to the set of information.

- **Discipline**: The discipline is a specific area of expertise within a general subject. Such concept is already considered in the current work and it covers a key role for the correct definition of the collaborative infrastructure.

- **Environment**: Natural conditions and induced conditions that constrain the design definitions or operations of a product. Such definition refers to all such entities that are not directly part of the system and their identification allows a clear understanding of the relationships between the system itself and external entities. The boundaries that characterize the interaction between various entities can change on the basis of what is defined as system for that specific case. The same set of entities can in fact be differently termed on the basis of the considered boundaries between the system and external environment. Generally the external environment can be confused with the whole environment where the system is also included. This concept must be clearly identified. In the current work the term environment can also be identified with the external environment. The union between the system and the external environment represents the whole world (basically what can also be identified as the whole environment) while the system must not be confused with the external environment since it is not part of such entity (considering the differences just introduced).
  The environment can be basically considered as an actor but this definition does not necessarily imply that it belongs to the system. A system contains a collection of items but there are no constraints on which items belong or not to the system itself. This depends in fact on boundaries features.

- **Function**: A function is defined as the intended effect of a product. Such concept is mainly related to the functional analysis activity that often comes before the hardware and actual component selection. In particular such process describes completely the functions and their relationships, which are systematically characterized, classified and evaluated.

- **Item**: An item may be more generally a product, a service or an actor. Such term has been considered to better formalize and describe the concepts introduced within such analysis (e.g. systems, product, service, actor, etc.).

- **Mission**: A mission is basically defined as a set of tasks, duties or functions to be accomplished by an element. This definition can be scaled on different levels on the basis of the object that is currently considered. The overall system has its own mission which is different from the one related to a specific component of the same system.

- **Model**: A model can be basically defined as a physical or abstract representation used for calculations, predictions or further assessment. A model can also be used to identify particular instances

of the product e.g. flight model, referring in this case to real object that however do not represent the exact system but is used to assess some specific behavior. In certain cases it is also possible to consider the definition of simulation models. In particular by simulation models it is meant here both data models, e.g. geometrical model of a system, and behavioral models, e.g. the algorithms representing the behavior of a component or environment expressed in a high level programming language. A model normally (but not always) has inputs, outputs and internal state variables and constants.
A generic model represents an entity (e.g. a power distribution network) that can be configured to represent any instantiation of that entity.
Although generic models are a powerful concept, they can become over complex and it becomes more effort to configure a generic model than to develop a specific model from scratch.
Depending on the context, models can be classified according to their fidelity, their domain or their modeling technique.

- **Modeling technique**: The modeling technique identifies the method used to analyze and describe the behavior of a model. Common techniques can be represented by the following types:
  - Physical (electrical, mechanical, etc.)
  - Behavioral
  - Functional (with respect to external interfaces)
  - Geometric

  Other different types can also be considered on the basis of the specific modeling needs.

- **Performance**: In the current work the performance is defined as a quantifiable characteristics of a function and it allows to evaluate the behavior of certain elements, paving the way for the comparison between two different entities.

- **Process**: Such concept can be defined as the set of interrelated or interacting activities which transform inputs into outputs. Inputs to a process are generally outputs of other processes.

- **Product**: A product is defined as the results of a process and in this terms can be represented basically by services, software, hardware or processed materials for example. Following such definition the concept of product can be related both to tangible (physical system) and intangible (service) entities. It can at least be associated to a collection of tangible object at one level.

- **Project**: A project is basically represented by a set of coordinated and controlled activities with start and finish dates, undertaken to achieve an objective conforming to specific requirements, including constraints of time, cost and resources.

- **Requirements and Specifications**: A distinction between the terms related to requirements and specifications will be useful for the following sections. Such terms are in fact widely used in the field of System Engineering and a clear description of their meaning can help to avoid possible misunderstandings. Their function is quite similar but the specific meaning is associate to the processes of design and analysis of a product.
  Requirements are what your product should do, the specifications are how you plan to do it. The requirements represent the application from the perspective of the user. The specification represents the application from the perspective of the technical team.

- **Service**: The term service refers all the intangible entities that can be involved during an activity or an interaction between some actors. In particular such definition can be associated to the action of that characterizes the interaction between actors during a specific scenario. The related entity is basically intangible both on macroscopic level and microscopic one.

- **Simulation**: The simulation concept refers to a run of scenario in a simulator with a simulated start- and end-time. During the simulation events may be injected into the simulation by the user, a script, external hardware or another simulation. Such definition more generally can be extended to run that not necessarily involves the time dependence. The most part of simulations are currently defined in the time domain but that is not an absolute law since there are other specific types of simulation that do not involve directly the time dependence. The dynamic evolution of current in an electrical power subsystem refers to the first type of simulation for example. The simulation of a structure response loaded with a set of stationary forces and moments represents another type of simulation. The main purpose of simulations is to foresee the behavior of a system interacting or not with an external environment, providing useful data to evaluate the responses before some specific scenarios occur. In such definition the time role is not necessarily present since some particular computations are not directly involving time dependence.
  Simulations can be used to support analyses since they provide the results needed to achieve a certain response but are but are not necessarily required by a specific analysis. In the same manner a simulation can be done without a direct connection with an analysis, since it can be formally used for other purposes (such as representative purposes for customer/supplier for example).

- **System**: Set of interrelated or interacting functions constituted to achieve a specified objective. Such concept must not be confused with the product term since a product can be defined as a single entity (able to be identified as a single entity) while the system requires however the presence of interacting entities. In particular the same entity can be described as product or system but on different levels since in the first case the main attention is on the whole entity as a unique element while in the second definition the main focus is on the various entities and their relationships. Such two concepts are however so similar that they are often considered synonymous. A clear distinction between such two definitions can help to better organize and formalize the overall meta-model structure. The actual benefits that can be achieved with the separation of such two terms is not often fundamental in the applications that have been developed so far.
  A system can be defined more succinctly as a collection of items and the related interactions. System boundaries allow to clearly identify which items belong to the system and which do not. The related classification strictly depends on the specific case and related scenarios.
  System concept must be clearly distinguished from the external environment since it generally includes all the entities with which the system interact with. The system is affected by external environment as also in turn it can influenced by system behavior. This distinction is fundamental to understand the boundaries for the context under evaluation since such identification allows to characterize the occurring interactions. From this viewpoint the external environment do not belong to the system but its role is fundamental to model system interactions. Space applications deal often with the definition of Segment and the use of this definition is fairly widespread to such an extent that ECSS standard included it. In particular the segment is defined as a set of elements or combinations of systems that fulfils a major, self-contained, sub-set of the space mission objectives. Such definition is however not so constraining since the entities considered in the related definition can be also considered as systems their own. Segment is basically a convenient representation of a complex space system that in turn involves other systems. In this way such single system can be termed as a space segment but what it means is basically the same since it is only a matter of scale. A segment can in fact be considered as a system as any other one but in the context of complex space application this distinction can help to manage all the data involved. Examples of segment are represented by Space Segment, Ground Segment, Launch Segment and Support Segment.

- **Use case and scenario**: Simulation scenarios must be distinguished from the definition commonly accepted for the description of interactions in the field system engineering. In particular the concept of scenario must be well understand with respect with use case

  - Use case: defined as a group of scenarios linked together by a common user goal.

- Scenario: defined as sequence of steps that describe the interaction between a user and a system.

The development of a system is generally characterized by the identification a set of goals that are basically derived from requirements coming from the customer and project statement. The main purpose of the preliminary phases is represented by the definition of use cases and related scenarios that are first conceptually elaborated and then detailed as the project proceeds. Each system goals are generally related to one goal which in turn is connected to different scenarios. The same use case is conceived to globally represent the interactions that may appear to achieve a desired goal. On the other side the scenario defines the difference sequences that may occur during the interactions between users and the system for the achievement of the same goal. In this case the goal is the same but external situations, initial or boundary conditions can affect the time evolution of events and users roles (such as activities pursued within such scenario). Scenario concept focuses on the temporal and different situations that can occur for the same use case. For example achieving the comfort temperature of a building represent the same use case since it is related to the goal of reach a certain temperature for the comfort of people. The same use case can however be accomplished in different manners (i.e. scenarios) on the basis of the current initial and boundary conditions. In the same previous example the operations that a single user can do to maintain a certain comfort temperature depend on the initial temperature. If the initial temperature is lower than the comfort one, the set of actions and system interactions are different from those that come out when the initial temperature is higher with respect to the comfort one. The same reasoning is valid when the system component that allows reducing environmental temperature (because the actual temperature is higher than the comfort one) breaks down. Such situation represents however another scenario for the same use case (i.e. achieving the comfort temperature). Scenario description basically contains all the information about who does what and when, expressing the sequence of actions between the involved actors and system components. The concept of scenario can be expressed in slightly different manner from a simulation viewpoint but the meaning remains basically the same for the purpose of the current work. In this case the scenario can be reported as a particular initial configuration of a simulator and sequence of events to represent a particular part of a mission e.g. launcher deployment, eclipse operations, cruise phase. The identification of scenario boundaries depends strictly in the case that is under evaluation.

- **Validation**: Validation can be considered as the process which demonstrates that the product is able to accomplish its intended use in the intended operational environment. In this context the verification is a pre-requisite for validation.

- **Verification**: Verification is defined as the process which demonstrates through the provision of objective evidence that the product is designed and produced according to its specifications and the agreed deviations and waivers, and is free of defects. Verification can be generally accomplished by one or more of the following methods: analysis (including similarity), test, inspection, review of design.

- **Workflow**: This term is used to define the process within which the analyses are executed. The workflow describes in fact the time dependence of the involved analyses, ordering the flow between the various elements. Conceptually if the dataflow defines which variable of which analysis is connected to which other variable, the workflow tells us when a certain analysis must be performed. Such concepts must be clearly separated to correctly understand the relationships between the analyses that characterize a multidisciplinary simulation.

In the context of a simulation environment it is also important to clearly define a shared meaning for the words: variables, constants and parameters. Such terms are defined in the following lines apart from the previous definitions since their roles are strictly connected. The following definitions are partially derived from the Modelica conceptual classification since first the related expressions are clearly formalized and

secondly they pave the way for the integration with an object-oriented environment (Modelica based for example).

- **Variable**: The variable is generally defined as a quantity that can be used to describe the behavior of an object, providing for example the values related to the states of the component itself. Such definition does not necessarily lead to the fact that the quantity is always characterized by a variability. Under particular condition a certain variable may not change its value over time for example. Its variability is in fact strictly related to the specific simulation and it is generally different from case to case. From this point of view a variable defines more properly a quantity that can potentially change during a simulation. This definition includes both the input and output variables since the related terms are expressed in the same way. Variables represent basically the quantities that with constants and parameters allow to compute/simulate the behavior of one or more elements, providing the numerical values required from the equations/functions available in the code for example.

- **Constant**: A constant can be basically defined as a quantity that does not change during a simulation and its value is not directly accessible by the user. Such term includes all the physical quantities and constants that cannot be freely modified or chosen by a designer or analyst. Their values are however needed by the equations/functions of a particular code to compute or simulate the response of the system under evaluation. From this point of view its role is basically the same of a variable while from a modification perspective the access is quite different (the user cannot modify this values on its own or through other similar mechanisms). Constants are used to define all such values that cannot be arbitrarily chosen by the user.

- **Parameter**: Parameters are defined within this context as those values that are required like constants and variables to solve/compute the equations or functions of a particular simulation. In particular such term identifies the quantities that does not change during a simulation but can be set by the user or however can be modified on the basis of a particular choice. They refer to those values that can be chosen at the start of a simulation but cannot be modified during the simulation itself. They are defined once before the simulation starts but can be changed after the execution to set another computation. The distinction with the constants values are mainly related to the fact that the parameters can potentially modified by the user/analyst on his/her needs. In fact a parameter can remain constants during a set of different simulations but this does not means that it can be included within the constants definition.

Another important distinction has been done between the concepts of option and alternative. This consideration will be reintroduced in the following section to clarify some conceptual choices for the framework. In particular such distinction is introduced to take into consideration in a better way the management of design changes and configurations. The term alternative refers to the design object that theoretically substitutes another one. In this case a certain element must be present and there are different alternatives that can be considered. In the current work the term "option" refers more properly to a design object that can be present or not, depending on the specific case. From this perspective an option is not related to an object that must be present. Such definitions is currently not clearly defined in the available standards and they are not yet formalized. The concept of design option is however present in some formalization works but is approached differently on the basis of the related organization or research initiative. Within the proposed approach such distinction has been considered quite important for a well-posed management of design changes.

The previous introduced definitions are then used as starting point for the formalization of the concepts used in the current work. In particular the concept developed in the context of Virtual Spacecraft Design project [57] have been considered and partially modified to take in to account some of the aspect that has been approached. The main concepts related to the topological definitions are reported in the following section. Such terms are mainly used to formalize system physical architecture, allowing the description and characterization of the relationships between product components.

Such definitions have been introduced to support the conceptual data structure about the model-based

methodology that has been evaluated. They are not the only available from the ECSS technical memoranda since other expressions and concepts are expressed with more details. In particular they are selected on the basis of the concepts that are considered for the current study. Such concepts are then properly used within the data model to define the related classes that are then elaborated to build up the Ruby on Rails classes used to implement the design framework.

6.2.1 Topological definitions

The following concepts have been defined to clearly represent system topology on the basis of an object oriented approach. Their meaning is strictly related to the conceptual data model used to support design activities at system level, providing useful capabilities in the context of the proposed infrastructure.

Element Definition

The term *Element Definition* used for the definition of the meta-model refers to the conceptual representation of certain object. It includes all the features that characterize the element that has to be represented within the system architecture. It contains the definition of all the attributes, methods and main characteristics for the represented element. It is substantially a class from which the objects used for the design of the system inherit all the properties. All the object that are linked to the same definition change consistently all their properties once the *Element Definition* is modified. This concept is particularly related to the definition of modularity and reusability of the design object.
In particular an element is defined once (the *Element Definition*) and then can be used (the *Element Usage*) nay number of times in an architecture (which may contain a hierarchical decomposition) about the system of interest. The *Element Definition* and *Element Usage* structure represent together the architectural design/composition /decomposition of the system of interest. The combination of *containedElement* property and *referencedElement* property of the *Element Usage* a hybrid product tree can be represented. In this case both the logical and concrete (also known as physical) architecture are combined. The containment relationship between the *Element Definition* and *Element Usage* instantiated from it allows to inherit some properties. In particular the *Element Usage* becomes automatically a member of a certain category if that one is assigned to the relative *Element Definition*. From this viewpoint it is possible to say that the *Element Definition* concept reflects the *Block* concept typically used in the context of the ontology of OMG SysML.

Element Usage

The term *Element Usage* identifies the instance object obtained from a certain *Element Definition*. It inherits all the feature assigned to the relative *Element Definition* (for example all the related mass properties). It represents the actual usage of certain *Element Definition* within a precise context. The same *Element Definition* can have different *Element Usage* but at the same time a particular *Element Usage* can refers only to one *Element Definition*. Once a certain *Element Definition* needs to be defined within a particular context for the definition of another *Element Definition* then the conceptual object must be instantiated. The *Element Usage* represents an *Element Definition* once this has to cover a certain role within a design architecture.
Both *Element Definition* and *Element Usage* can be typically related to a top-down viewpoint in the context of design process. In particular the properties are defined as they theoretically must have in the actual project. They are assigned for design purpose but not reflect necessarily the actual properties of the system (as this start to be produced and realized).
This object identifies in particular those elements that are represent an usage of a certain *Element Definition* in the context of an higher level Element Definition that contains that usages. An *Element Usage* as defined has one and only one *Element Definition* that contains it.

Element Occurrence

The term *Element Occurrence* instead covers the role mainly related to the design process for the phase of simulation and analysis. It represents a particular element or component of the system as it is computed. This definition is addressed to those phases related to the simulated/computed activities. In this case each *Element Usage* is directly related to one *Element Occurrence*. This concept reflects mainly the design process from a bottom-up perspective. In this case the element characteristics reflect those coming from the computing or simulation process. In this case the properties are inherited from the object not as a priori defined (for example as in the case of *Element Definition* and consequently for the *Element Usage*) but as computed. This object identifies a specification of a reference to a specific occurrence of an *Element Usage* in a fully expanded tree of *Element Definition* and *Element Usages*. The identification of a particular occurrence can be derived from the root *Element Definition* and the ordered list of the subtended *Element Usage* references. The concept expressed with this object can be referred to the “deeply nested connector” definition in the context of OMG SysML v1.2.
Element Occurrence concept has been conceived to directly generate and identify all the possible instances that can be present on a certain system, exploiting as much as possible the use of *Element Usages* and *Element Definition* and their information, reducing the time required for the characterization of elements already define.
For example if a motor-wheel has been defined (*Element Definition*) as the assembly of two *Element Usages*: the motor and wheel (which in turn are Element Usages of the motor *Element Definition* and wheel *Element Definition*), then six motor-wheel can be instantiated as *Element Usages* for the locomotion system definition of a rover. In this case it is not necessary to redefine the contained elements within each motor-wheel. The six motors and six wheels represent respectively the six *Element Occurrence* of the motor and the six *Element Occurrence* of the wheel. Their generation (as *Element Occurrence* entities) is automatically obtained through the information available from the *Element Definition* of the single motor-wheel (which also contains the reference to the *Element Usage* contained).
The *Element Occurrence* class has been also been defined to take account for a better direct integration with the data coming from the various disciplines. This object in fact can be used to properly related the properties coming from the models already defined for certain elements and the information contained within the main *Element Definition* entity. This approach partially follows the concept that can be identified two main design "direction" during system development where an ideal model definition (by for example the people working at system level) can be concurrently integrated with an already implemented model (for example coming from structural design process). The main idea is to consider both such information at the same time, providing an useful perspective to better manage and monitor the system design as the development proceeds and the maturity level of architecture becomes more detailed.
A description of such concepts is reported with more details in other sections of the present work.

Element Realization

The *Element Realization* concept refers to the system element as realized. In this case the properties (for example the mass properties) are not inherited from a conceptual definition (the related *Element Definition*) but are obtained from a measure process for example. In this case the characteristics are typically provided by a bottom-up process. In particular the object properties that is identified with the *Element Realization* definition can be considered as measured. These properties and also other element characteristics are those closer to the actual product. This class has been introduced within the conceptual model to include the possibility that some design elements are already realized during the development phase. In particular this object models those elements that are effectively produced at the time the design solutions are evaluated. This situation can be encountered when some elements produced are reused and their definitions can be built from the available information still from the early development phases.
The conceptual basis of the present work are strictly dependent on the definitions just introduced since they are correlated with the theoretical infrastructure on which the framework has been developed. Figure 6.1 can help to better understand the role of each class that has been considered, providing a better explanation of the contexts with which such objects can be linked.

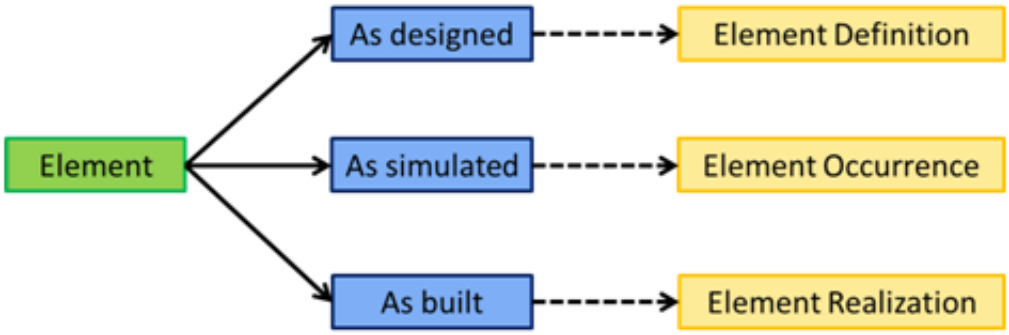

Figure 6.1: Summary of the elements conceptual classes and related modeling context.

The *Element Usage* class is not reported in figure 6.1 since the role of such concept is close to that covered by *Element Definition*. Both this classes are in fact related to the definition of an ideal model of the system or components under development. They can be grouped within the context of defining an object as designed, representing the theoretical target towards which the final design is addressed.

Element properties

Another important feature to consider during the definition of element properties is related to the fact that they can be differently managed on the basis of their relationships with parent object. In particular the proposed infrastructure is based on the inheritance of all the possible properties from the *Element Definition* to the related *Element Usage* but not all such quantities can be handled in such a way. For example some properties as the mass can be directly inherited from the *Element Definition* to the *Element Usage*. In this manner once the mass is defined within the *Element Definition* then the *Element Usage* must have the same mass. If that is not still true it means in fact that the a new *Element Definition* is needed to define the related *Element Usage*. Not all the properties are defined in such way since some quantities are not directly inherited from the *Element Definition* type but they depend on the specific instance of the *Element Usage*. This case allows to model all such properties that can be related to a specific element but are not directly derived from the corresponding *Element Definition*. An example of such entities is represented for example by the vector position of an element that defines the position within the parent element (geometric position or center of gravity position) and it cannot be derived from the related *Element Definition*. This property in fact changes among *Element Usages* that have the same *Element Definition* since it depends on the relative position within the parent. Such example shows how not all properties belonging to an object instance can be derived from the *Element Definition* but they are however fundamental to fully characterize the element.

6.3 Conceptual framework philosophy

In the next sections the main features of the proposed conceptual infrastructure are provided, focusing in particular on the relations between the concepts and actual objects. Framework philosophy is fundamental for the correct definition of the involved concepts and the formal descriptions of the elements in the previous lines is then used to clearly formulate the overall methodology. A clear formalization of such terms must not be undervalued since often different people work together on the same project and the same word has different meanings with respect to the person that is currently using it. Such situation could be the starting point for a set of problems and misunderstandings that can arise in other phases. It is important not just the definitions themselves as the fact that they must be globally recognized as shared definitions.

6.3.1 Conceptual meta-model of the proposed methodology

The main core of the conceptual architecture of the current work has been developed starting from the definitions, the classes and their relationships available from the current ESA standard ECSS. In particular the ECSS-E-TM-10-25A and ECSS-E-TM-10-21A technical memoranda have been considered as reference for the overall evaluation of the current methodology and the related integration. All the formal definitions used in the current work can be found within these technical guidelines. Some of the considered concepts are well explained and described in the following sections. This brief introduction allow to better understand the overall methodology and the choices concerning the development environment.
An important phase during this study was represented first of all by a deep analysis of the current design methodologies with particular attention to the people, processes and tools involved (as already underlined by the previous consideration regarding the different resources involved within a system design development). The correct characterization of the people involved within a certain process, the skills required to obtain a particular result and also the tools that can be developed to reach certain objective are all fundamental activity for the right generation of a useful solution. The generation of something that can even be a powerful tool but it is not well suited for the people that have to use this one (for example because the time required for the training is too long), is not a smart choice.
An important phase of this study was then devoted to a clear understanding of the process involved in the system design before a methodology and tool was proposed and analyzed. This process has been characterized by a series of alternative approaches for the integration of MBSE methodologies within design process, indirectly considering also the actual implementation of such approaches with a correspondent tools interfacing (since it is also important to understand the actual feasibility of the proposed approach). This analysis allow also to define which person accesses which resources, considering that different domain roles are involved on the same project but not all have the same access credentials to edit or delete something.
The following phase was represented by formalization of the meta-model structure on the basis of the initial considerations of this study and the main characteristics of the modeling architecture. The modeling environment plays a key role in the definition of the main features of the proposed analysis framework since it defines how all data and information are stored and exchanged. The conceptual formulation for the integration between a modeling context (directly related to the representative definition of the system) and analysis capabilities (related in particular to the feature provided by external solvers) requires first of all a clear understanding of the modeling meta-model. The meta-model integration with the concepts coming from the proposed analysis perspective has been introduced once the modeling conceptual architecture has been evaluated. One of the principal activity was represented by the evaluation about the feasibility of such integration, identifying also the potential improvements that such approach can directly introduce within a design process. The objective is to understand in fact if the proposed methodology can actually provide tools and services that support the work of engineers with different backgrounds but working on the same project. It is fundamental to understand how, where and when such MBSE approach can be integrated during the development process, avoiding the definition of a framework that is not easy to manage and that is completely away from the well rooted approach that characterizes the traditional design phases. The idea is to provide a conceptual model that allows the definition of a flexible environment for the investigation of design process, integrating some capabilities such as the set-up of models simulation and analysis.
The implementation of the conceptual elements starts from the meta-model that characterizes the representative definition of system and additional objects and classes are then added with the final purpose to manage the various scenarios that have been considered in the first part of this work. The objective pursued with such a study is the demonstration of the capability to manage the scenarios defined as reference cases to evaluate framework effectiveness, showing how the proposed methodology can face real engineering issues. The previously example scenarios are in fact modeled considering real design situations, exploring in particular some of the possible configurations that can be evaluated within a project.
First of all the definition of meta-model concepts has been characterized by the introduction and definition of classes to integrate with the ones mainly related to “representative” model of the system. Within this

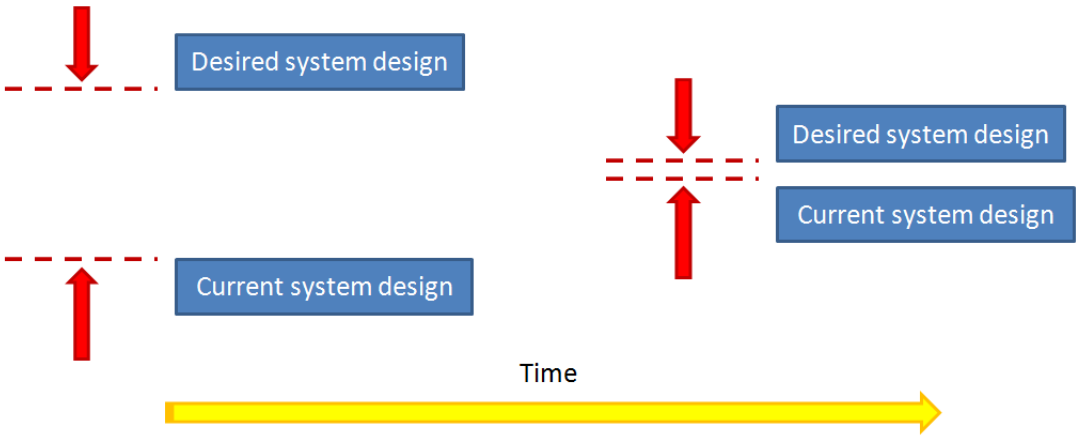

Figure 6.2: Conceptual relationships between the modeling activity for desired and actual system design.

context classes for the definition of analyses, simulations and the related code runs has been introduced to proper manage the integration with the already defined classes. The overall architecture of such conceptual formalization is based not only on the definition of the involved classes but also on the relationships that link each other. The proposed definitions have also the purpose to cover the majority of the current domain-specific design processes with eventually minor changes in certain specific cases. The proposed methodology shows interesting behavior with respect to the formalization of design processes, providing useful base for the standardization of the information collected from different domain-specific environments (with greater emphasis on those that are characterized by analysis activities).
The design process follows the conceptual approach proposed and under evaluation concurrently with the modeling framework which main features are represented briefly in figure 6.2.

A desired system design represents generally the target of modeling and analysis activities and as the development process proceeds the gap with the actual design decreases. The initial desired system design may not totally match with final one since during project evolution some requirements and customer needs may slightly change. Similarly some problems may come out as the system design becomes more detailed and a configuration modification is required to satisfy other boundary constraints. At the same time the current design can be considered as a constant evolution towards the requirements and specification that are modeled on the basis of the desired behavior for the overall product. This perspective can be related to a top-down view for the definition of the desired design and a bottom-up approach that shows the actual design. In this manner it is also possible to better figure out the main roles of all the people working on the same project. In particular system engineers are often involved on the phases as the definition of design solutions (which still need to be analyzed for example) while other domain-specific engineers are responsible for the implemented models (with analysts role). This conceptual division should not be understood in the strict sense since also analysts may propose design alternative solutions (in the case some requirements are not satisfied) for example. In the same manner also system engineers may be involved directly in specific modeling activities.
In this way the proposed methodology try to clear define the boundaries between the activities that studies and propose a solutions and those that are instead devoted to the verification of such choices on the basis of the implemented models and analysis. These concept are partially visible in the structure of the proposed framework and meta-model classes. The main idea is to clearly highlight the fact that there is a desired system design (associated with the central system model and managed within the main system repository) and a current system design (associated in this case to the analysis that are under development as other domain specific models which are instead managed in the related domain specific repository). This frame allows to organize all the information in a better way, providing useful instruments to compare the actual completion level (on the basis of a shared development schedule for example) and identifying at the same time the areas that need more resources.

6.3.2 Analysis and simulation meta-model concepts

Some efforts of the present work have been addressed towards the definition and formalization of such object needed for the integration of analysis and simulation concepts within the modeling framework. As in the previous sections a conceptual analysis has been done for the identification of the required objects, their attributes and methods. In particular the approach proposed has been developed considering some of the inputs available from actual engineering design problems. This analysis has tried to figure out what activities cover a fundamental role during such design processes, identifying all the elements that can be recognized as common. Once a list of common elements has been defined the following activity has been addressed to the description of the concepts that allow to formally represent the cases considered initially. This reverse process has been done to understand if the data structure and meta-model classes are properly representative for problems similar to those considered as starting point for the proposed framework. The proposed approach for the management of analysis and simulation starts from the definition within the meta-model of two different objects with two different purposes. This difference has been introduced to properly manage the references to files and resources involved during the design process, above all with particular attention to the amount of data that must be processed in the more detailed phases. This feature is mainly related to the choice of store files on server side or properly map their links (file-system paths about the related resources). Whichever approach is chosen is important to understand that this one is strictly related to the following implementation for the files storing and resources links (for example in the case some results files need to be committed).
The previously consideration are particularly important for the implementation of a multidisciplinary survey that in fact requires a well-defined simulation environments to properly manage the available information and to run without problems. The formalization of such process must be clearly described since it represents a key-role element for the framework under evaluation.
The considered integration about simulation environments and modeling framework has been conceived to support the design activities from system perspective with the capability to provide such service with a web-based architecture. This capability should not be confused with similar multi-domain simulation environments conceived to bound different simulator on different machines (distributed on a network) as for example HLA (high-level architecture) or other similar protocols. This one is in particular a general purpose architecture for distributed computer simulation systems and using such protocol, computer simulations can interact between each other with no restrictions on the computing platform used. The overall interaction is managed through the Run Time Infrastructure (RTI) and all the involved simulations must however follow the pattern specified by the protocol itself. In this way all the computer simulations can communicate and all the executions are called in the right order on the basis of a synchronized process. This simulation integration requires however that all the involved codes are already built and updated to HLA protocol and for this reason it is not well suited for the management of models that are continuously changing (as in the case of design process). HLA represents a good choice for the integration of simulations already validated and working across computer platforms and in particular when a network communication between the various models is required. The design process is often characterized by models not fully implemented with respect to some aspects and a more flexible approach for the integration of simulations represents a good solution.
Meta-models includes also the definitions about the operative modes and scenarios regarding the system under development but they are not directly considered within the conceptual classes for the simulation set-up. At the end all the concepts and meta-model formulations have been used to implement the features related to the modeling framework but a more detailed description of the framework developed are reported in the next sections. The main purpose is in fact to evaluate the correctness and effectiveness about the main modeling and design conceptual architecture. From this viewpoint the developed framework should be considered as a demonstrator for the model-based methodology currently under investigation.

Concepts from other research initiatives

Both OCDT and VSD projects approached the conceptual definition related to analysis and simulation also if but they introduced classes and concepts slightly different from each other. In particular OCDT initiative considers the concept of *Rule*, *Iteration* (strictly related to Concurrent Design Facility), *Relation*, *Parametric Constraint*, *Option* and *Design Method*. The iteration concept in this case is strictly related to the feasibility and trade-off studies that are performed within the context of Concurrent Design Facility. In this context it represents an iteration in the process of developing an engineering model. Rule and relation have been mainly conceived to specify the relationships between categorizable things within the data model. The parametric constraint class is instead designed to specify a relation that consists of a parameter (that acts as variable), a relational operator and a value through equality or inequality constraint. The option class represents a potential design solution for the system of interest. It is basically a design alternative that can be compared with other ones to perform trade analyses for example. This concept is basically conceived to collect options at system level and it is not mainly conceived to model options at a lower level. Conceptually the design options are not so easy managed as the system complexity increase with this approach. Options trade-offs can in fact involve design activities with elements at a more low level (component level for example) during the detailed phases of a project. Under these conditions the management of different low level options can become more difficult to control and correctly perform. The methodology proposed in the current work approaches such problem in a different manner introducing a slightly different conceptual definition. In particular a more detailed overview will be provided in the next sections.
The main concepts related to analysis and simulation that are coming from Virtual Spacecraft Design project (already introduced in the first part) can be summarized in the following ones: *Analysis Execution*, *Analysis Design*, *Analysis Model*, *Analysis Run*, *Analysis*, *Analysis Case* and *Analysis Result*. They are currently implemented within the infrastructure that arise from VSD research initiative and are used to validate the related data model.
In both cases, considering in particular the initiatives related to VSD and OCDT projects, the previously introduced concepts are not fully validated and they are currently under investigation to understand the possible improvements with respect to actual engineering project.

6.3.3 Design Variables main conceptual definition

One of the most important feature of the integration of design variables within the modeling and analysis tool is represented by the correct definition of such elements. Such process must be supported by a well formalized conceptual meta-model for the design variables. The related class has been defined considering mainly the sizing and design process where such object have to be inserted. Starting from this perspective the main attributes are identified and included within the definition. In particular the variable properties selected in the context of the proposed methodology are represented by the name, the description, the type and the features related to this one. Other attributes as the nominal value and the possibility to consider such object as operational or not for the analysis to be considered. In the case of closed variability for the design variable under evaluation the related value chosen can be directly represented by the nominal one.
The purpose of the attribute related to the operational status of the design variable under consideration has been introduced to provide the capability to choose if certain design variable is however under evaluation or of it can be managed properly for the definition of a particular survey (trade-off, optimization, uncertainty quantification, etc.). This attribute in particular can be denoted with ACTIVE attribute, referring directly to its status in this way. This property can be managed for example through a Boolean value that allows to understand if the parent design variable is currently under evaluation or if the related value has been fixed on the basis of the already done analysis. The idea that has animated such a choice is represented by the advantages to introduce an object for the monitoring of the open design variables. The main benefit is also represented by the possibility to proper trace the changes for the single design variable.
The name and description attributes do not require particular explanation about their meaning in the con-

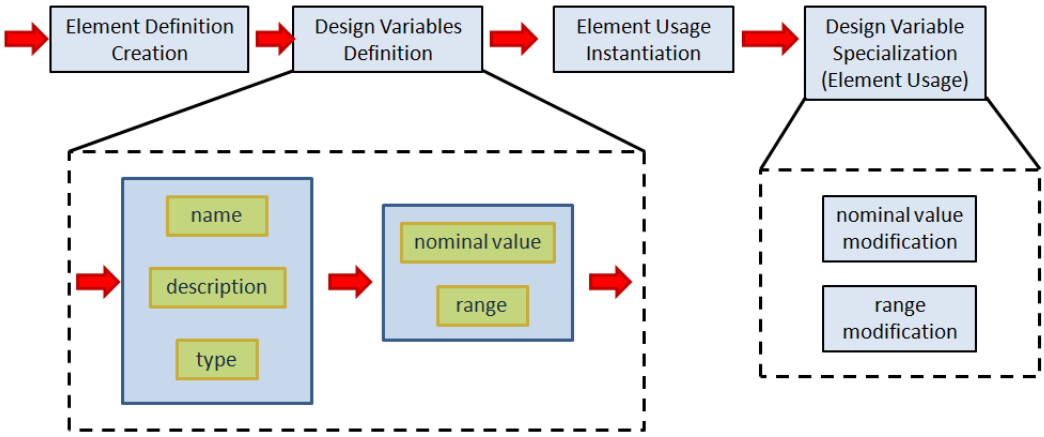

Figure 6.3: Conceptual view of an example definition process related to design variables.

text of design variable. The type attribute has been conceived to better manage the possible automated processing once a simulation tool is integrated. Specific range settings and further elements can be introduced starting from the type definition, reducing the possibility to introduced erroneous information and improving the design process formalization at the same time. For example once the design variable as been defined as continuous then a maximum and minimum values will be provided to define the range the parameter belongs to. In the same way can be managed discrete variable, enumeration ranges or statistical parameters. The nominal value is instead introduced to guide the settings for potential surveys (as initial value) but also for example to store the value coming from analysis once this one has been formally fixed.

For the same *Element Definition* it is possible to access different classes of design variables which introduction will be better understood with the following examples and scenarios considered as reference cases.

The current meta-model formalization considers three main categories for design variables. In particular within the same Element Definition component it is possible to define one or more individual design variables, one or more groups of design alternatives and one or more groups of design options.

Alternatives group stands for a set of solutions/configurations that are mutually exclusive between each other while Options group identifies a series of objects that are not mutually exclusive between each other. This conceptual classification is the one used in all the present work. Both alternatives and options group are mainly conceived to manage physical components inside parent one as will be better explained in following examples referring to real problems.

A conceptual flow for the modeling activities related to design variables is reported in figure 6.3.

The figure conceptually illustrates an example of process flow for entering individual design variable within a specific *Element Definition*. The design variables definition can be done as all the other available activities in the context of the just created object. Name, description and type attributes are defined in this moment and are available as main characteristics for the Element Usage that are instantiated from this object. Nominal and range attributes are also defined in the same operation but they are treated differently since they can be modified once the *Element Usage* has inherited the first inserted values. In this way it is possible to model for example two element coming from the same *Element Definition* (and in this way the same design variables) but with the capability to take two different range values. This feature allows to face design problems in a more flexible way, enhancing the possibility to configure object conceptually the same but with slightly different boundary conditions. In particular the main idea in this case is represented by the fact that the user is able to modify the range and nominal value independently among various *Element Usages* that are all inherited from the same *Element Definition*. Once the nominal value of the design variable has been introduced for the *Element Definition* then the *Element Usages* that are instantiated from this one inherit the same nominal value but its change is not prevented since all the *Element Usage* are independent between each other. In the case the design variable must assume the same nominal value among the various *Element Usages* then such scenario must be modeled introducing

a specific constraints for the considered design variables. The *Element Usages* are in fact proposed as independent elements a priori but future implementations may consider the possibility to directly impose from *Element Definition* the fact that the nominal value of design variable is the same among the various *Element Usages*. The proposed meta-model and the related methodology for the management of design variables (conceived for the definition in the context of modeling environment) has been conceived to support the design phases of a system. For this reason is particularly important to clearly understand the role of such capability and avoid the potential generation of a over detailed set of information. An excessive amount of data must be avoided as a lack of information. The considered approach for the definition of design variable has been conceived to improve the exchange of data between disciplines strictly involved on correlated engineering problems. The main idea is based on the sharing of those design variables that strictly affect the design of different disciplines at the same time, paving the way for the building of a multidisciplinary design environment. In this manner it is avoided the possibility to share design variables that are strictly evaluated and investigated by a specific discipline. The classes defined with alternatives and options groups are introduced to better organize the available information in the context of design process. Both these groups must formalize a set of possible solutions strictly related to a specific system aspect. Options groups can conceptually be represented only by one set since the term options as intended do not affect the other elements. They are in fact independent from each other and one group of options for the individual *Element Definition* is enough to model own needs. The meta-model schema allows for the definition of multiple options groups also for the same *Element Definition* since a clearer perspective and a more structured representation Is provided in this way. A specific engineering domain can upload their options groups separately from groups coming from other department, ensuring a more readable representation of the information. In this way options groups can be organized and stored enhancing their belonging to thermal or structural domain for example.

The alternatives and options management approach does not consider the introduction of constraints at this level. Object constraints are defined separately from the design variables introduction procedure. This phase has the main role to define all the possible solutions and combinations between the alternatives and options groups introduced. Theoretically speaking all the data entered at this stage can represents a potential system configuration. The check about the correctness and feasibility of a specific variables arrangement is made a posteriori evaluating the constraints that are defined in another section. The rules that must be satisfied are evaluated through analyses and simulations of the system (or components) configuration under investigation. In this way the main purpose is to clearly separate the creation and exploration of design space from the constraints and requirements. The implementation of a priori determination of which configurations are feasible may be more efficient but shows some integration difficulties at the same time.

The *Design Variable* class may confused with the conceptual definition of another component property since during the development process theoretically all the features related to a certain object can be considered as design parameters. For example the mass related to certain component can be defined in the *Element Definition* object and then modified as design proceeds. Then this property can be defined as a design variable (in the same manner also all the other Element Definition properties can be viewed as design parameters) and for this reason the *Design Variable* class seems to be not properly correct in the context of the proposed framework, since it must be considered as a duplication of information. The Design Variable class has been mainly conceived to related the modeling framework with the multidisciplinary design environment. Such class allows in fact to better formalize the characteristics of the parameters that in this way can be integrated in the context of a analysis environment. The creation of component design variable based on *Design Variable* class must be animated by the need to share such parameter with other disciplines, paving the way for potential surveys that have to be done to assess specific investigations. At the same time nothing prevents the possibility to associate a design variable defined in this way to a component property previously defined. It is theoretically possible to substitute for example an optimal value (found through proper assessments) if this value is directly related to a certain component property. Such proposed approach has been introduced to clearly identify those design variables that are shared with other disciplines in the context of a multidisciplinary environment.

From all the previous considerations the elaborated concepts are summarized in the metamodel scheme

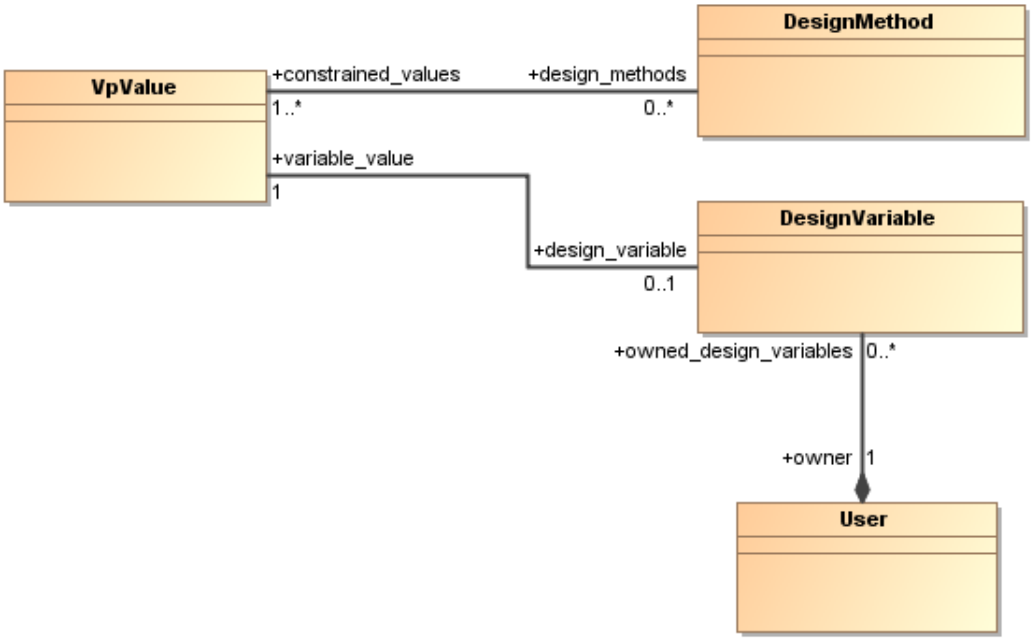

Figure 6.4: Metamodel association related to the *Design Variable* class.

and correlated with the other classes. In figure 6.4 is reported the section of the overall metamodel directly related with the *Design Variable* class. In the same figure 6.4 is also possible to see the *Design Method* class that will be however introduced in the following sections. In particular the association related to such class is currently implemented in a slightly different manner, taking into account in fact the mapping between the values available from the system model and the quantities of the design methods themselves.

6.3.4 Constraints and formulas management

A clear distinction between the concepts of verification methods and design methods must be done before more detailed considerations are introduced about constraints and formulas. Verification methods are generally used to control the correctness of design solutions, monitoring the current status of the project with respect to system requirements and specifications. Design methods are instead related to the generation or computation of certain system properties or design solutions. Such methods are used to support the design activities, providing useful instruments during the development process. They are applied to evaluate design data or also suggest specific choice among a range of possible ones. In particular an example of design method can be represented by a simple rule of thumb that can be used in the preliminary phases of a project to roughly evaluate an indicative behavior or value for a particular element. This concept will be strictly related to the definition of formulas. The framework data model considers also the definition of the two important concepts represented by the constraints and formulas formalization. SysML language provides useful diagram to manage both these elements (Parametric diagrams) which are basically considered at the same level. The definition of formulas or equations with SysML language can start by the definition of Constraint block that allows binding some parameters in a certain manner. Some model variables can be linked to such Constraint block once the related relationship has been defined. In this way a SysML tool solver (for example ParaMagic, Cameo Simulation toolkit, ParaSolver, etc…) can try to obtain the required values on the basis of the availability of inputs. In this case the solver itself tries to correctly assign the causality of the involved variables, assuming that the formulated relationships and available quantities lead to a well posed problem. In this context Constraints and Formulas concepts are not so different entities and can be managed substantially in the same way.
The proposed approach and the related framework face such definitions in different manner since these two objects are managed differently during design process also if they seem to be slightly similar. Constraints can be defined in a way similar to formulas, using basically the same semantics (number, operators,

etc…) for the definition of the related expressions but their evaluation has different purposes.
Constraints are generally evaluated a posteriori after analyses have been done while formulas are mainly used to evaluate something that is not known a priori. In particular formulas may be used to compute variables to assess constraints satisfaction. Both these object can however be implemented using a scripting code to model the related relationships and expressions. The language used for this purposes can be different as Modelica, MathML, etc… but the first one seems a well suited solution.
Constraint and formula expression more generally can contain algorithms implemented also with conditional operator but simpler relationships can be considered without losing the capability to well represent real problems. In this way the complexity level will depend on actual situations and specific needs. One implementation difference is related to the fact that constraint element can contain operator like equality or inequality sign while formula object is defined with equal sign. The evaluation of constraint element should return a response about satisfaction or not on the basis of the available values (a third response type can be related to the case where something is missing and the solver is not able to evaluate the related expression, for example providing a warning).
Following such considerations the related concepts are properly formalized within the data model. A constraint definition element has been introduced with the related attributes and associations. Constraints definition belongs to certain project and can be inserted and edited from an interface on the main page of modeling environment. Future improvements will consider the possibility of a direct association of such constraint element with the requirements. The single constraint is evaluated on user command and the results from this check rise from the current available values for the properties linked with the expression itself. In this manner the main idea is also to provide utilities to support the automated check of routine activities. For example it will be implemented some capabilities as the automatic verification of all the constraint related to the current project at the same time instead the verification of only one element at a time.
The definition of Constraint class has been conceived with the main purpose to provide editing capabilities for the code used to model the related expression/algorithm. The variables contained within the related expression are then properly mapped to the properties values of Element Definition for example from which maintain however an independent representation. In this way when a constraint is evaluated the related variables are substituted in the expression with the value currently present on the latest version of system model. The methods defined within constraint class are then also used to check for the correctness of the relationships, returning a feedback on the satisfaction or not on the basis of the available data. Editing activities and updating of already defined constraints can also be done to manage such kind of information. Constraint objects are defined as element belonging to project and not directly included within the components of the system since in this way different bond can be used on the same component in different project.
The definition of Formula concept is another important feature related to the design phase. Such concept has been introduced mainly to manage the "rule of thumb" expressions that are often used in the early phases of development. In particular this class has been introduced basically to model those expressions that can be evaluated by an external solver in a short time. These objects will not be linked to a particular project since they offers functionalities that can be reused on other design processes while constraint element are instead bounded to a certain system development (strictly dependent on the current requirements and needs). Such utility can also be implemented following the same definition of Constraints class as the use of Modelica code to evaluate the related expression. Analogous results can also be obtained through the use of JavaScript language with the advantages to load the required expression on client side, reducing the latency due to server response. This last approach can also be used since the possible computations are independent from the information collected from the server. They will be used mainly to support simple design evaluations, providing instruments useful for the identification of values consistent with the properties under development.
Formula concept must not be confused with a more generic capability that is natively embedded within the web based environment since there are standard simple computations that are quite common across different project (for example mass budget, power budget, etc…). The formulas that have been previously introduced within a project are available to other ones since the idea is to collect such information in a

common repository. These two concepts must be considered separately since they are addressed to two different purposes. While one is used to compute directly a specific system property (mass budget for example is used to evaluate the current state of the project with respect to mass property) the other is used to evaluate an output value physically consistent with the specified inputs (for example a simple solar array formula can be used to obtain different power outputs values on the basis of various array surface extensions).
The main characteristic of the formula definition is represented by the fact that the expression implementation is not directly dependent on properties specific to a particular project. The formulas are defined starting from a generic expression and then the related variables and parameters can be managed in different ways. The main idea is to evaluate formula outputs on inputs provided in different manner:

- Inputs defined through mapping directly with system model properties.
- Inputs defined through user provided values.
- Inputs defined both with mapping of system model properties that through user defined values.

The correct interpretation of the code content and then the identification of algorithm/expression are based on the right parsing of the script itself.
The constraint object is basically introduced to check the possible violation of user defined relationships between numeric values associated to system components properties. The main idea is to return only a Boolean response on the evaluation of constraint code with respect to the values passed as arguments. Such feature must be not confused with the results that can be provided by a simulation since the constraint verification requires only the evaluation of the correctness of the current values. A code similar to that implemented within a constraint can also be used to provide the result of a specific simulation item but in this case the final purpose is quite different. For example a Modelica code can be used to build a simple rule of thumb for the evaluation of some particular property in the preliminary phases and the same code structure can equivalently be used to define a certain constraint. The two concepts must remain separated since even if a similar Modelica code is used for simulations with low fidelity level in the first case, such code is however related to a simulation item.
Currently the constraint verification has been conceived to evaluate static variables through their substitution with the mapped parameters of related algorithm but more complex investigation can further be added in the future (for example considering the evaluation of constraints involving variables that comes from dynamical simulations). The required value for the evaluation of constraint are ideally all available from the system model but further development can include also the evaluation of parameters that depend on the run of external simulations. Such situation comes out when some of the required variables are provided by the run of a simulation item directly linked with the mapped property.
Design and dimensioning processes are often supported by simple formulas and rules of thumb that are used to rapidly give a preliminary idea and an indicative value of a particular quantity. This situation has been considered and the related concepts formalized at data structure level, providing another interesting feature with the definition of a specific class. In this manner simple functions and formulas can be shared among the people involved in a project and can be uploaded through a dedicated interface once a certain expression is not found. A library of related formulas can then be populated as an increasing amount of actors are involved in the process, reducing the time required to redefine expressions already uploaded and improving also the knowledge and information infrastructure.
The objects formalized basically as formulas are not bounded a project with reference to the relationships defined within the data structure. In this way such elements can be managed independently across projects providing useful capabilities that can be shared avoiding also the time consuming process of reinvent something that has been already defined. The same expression can be so used into another context mapping another set of component properties. A formula evaluation has been conceived to be mapped both to values already defined within the system as also through user-provided quantities not directly related to the that contained within the project (for example with the final aim to investigate the possible results of certain input). The correct evaluation of formulas code is obtained through a parsing of the contained information, providing the filed for the set up of the available input.

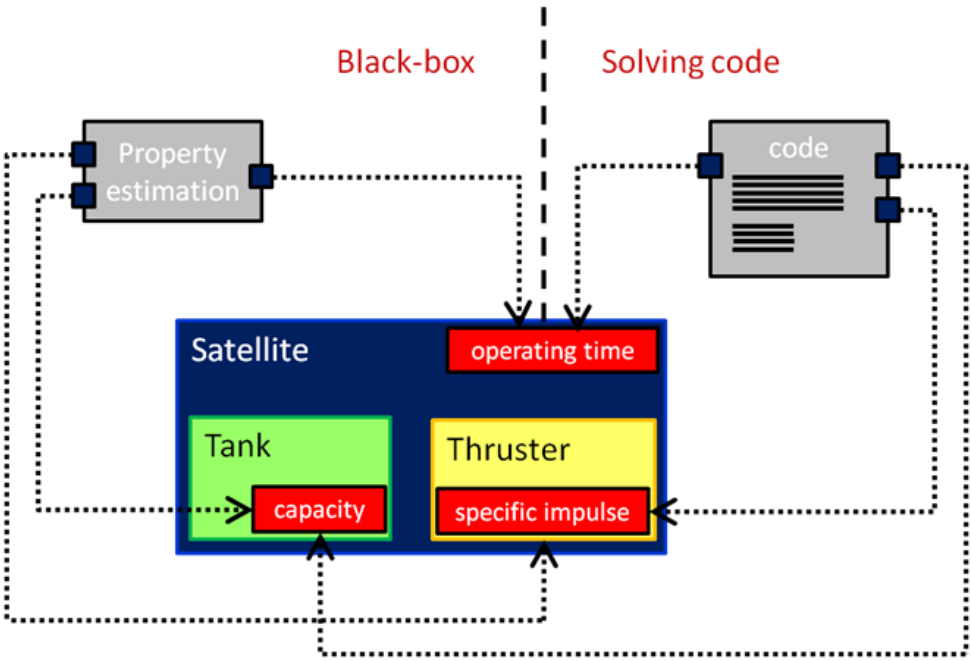

Figure 6.5: Conceptual view of properties estimation approaches.

An example about the main approaches that can be considered for the management of properties evaluation is reported in figure 6.5. In particular an example referred to the estimation of operating time property of satellite is reported highlighting the dependencies with respect to the properties of contained elements (tank capacity and thruster specific impulse in this particular case).

The concept of formula has been basically formalized in the class *Design Method* and all the previous considerations remain valid. The *Design Method* objects are used to model all such formulas and equations (including also the functions used to define the available "Rules of thumb") that are used to compute some specific values or properties. A set of values can be linked to a design method through a mapping between the related quantities (the corresponding ones available within the design method expression). In this way the design method remains an object "independent" from the values loaded within the model and it can be reused in other scenarios/components. The relationships between the actual properties and the corresponding "positions" within the design method is stored through the mapping object. The design methods have been conceived for the fact that they are used to run the computation of a certain values as output (it can be used to compute one or more output but it must be checked that the a certain value that is output from a design method belongs only to one method at time to avoid potential conflicts) and then they can be directly loaded/stored within the system model or not, depending for example on the user choice. During the development phases one or more input variables of a design method can change and the updating of the computed output values can be managed through different approaches. The update can be automatic on the input variables change or more properly controlled by the user through a user in the loop check. The final solution depends on implementation purposes but does not affect the main features of the conceptual model considered. The core computation associated to the *Design Method* can be implemented following different alternatives. If the computations are not so demanding they can be done through a Modelica based code (for example implemented with the algorithm section available) directly storable within the object itself. There are however no limitations on the fact that for complex computations are done through executables or external codes directly linkable to the *Design Method* class with the *Resource* object. In the same way also the *Constraint* class can be managed with complex codes through external resources.

With respect to the object *Constraint* previously introduced, the mapping mechanism is the same as for *Design Method* but their scope is quite different. In the case of *Constraint* object the mapped values are connected with the final aim to verify the correctness of the constraint rule. In particular the relate constraint rule is evaluated through the provided values and properties and it only checks if the expression is satisfied with no aim to generate an output to load or substitute with another one in the system model.

It basically collects the numerical values mapped with the quantities contained within the constraint and evaluated the expression itself. The result of such investigation will be the satisfaction or not of the rule considered.
It is generally possible that the evaluation of the quantities that must be provided to the constraint rule depends on design methods. Such event is foresees and happens for example when a certain quantity linked with the constraint is in turn an output variable that must be computed through a design method. In this case the computation of the constraint must pass first across the evaluation of all the quantities needed before the final investigation of correctness of the rule.
The concepts of *Function Model* and *Design Method* (more details are available in the appendices) show similar features that in certain cases can also overlap but they are basically conceived to model two different situations. In the case of *Function Model* the main aim is represented by the evaluation of system (but the same concepts applies also to subsystem or the individual components) behavior on the basis of the available data from the system model. In this case there no limitations on the fact that some properties or variables result from the related computations/simulations. The final scope is to simulate the response of the system and not the direct computation of quantities that can be used within the design itself. The *Function Model* wants only to show how a specific object (system, subsystem or component does not matter) behaves with no primary attention on the evaluation of specific properties. On the other hand the *Design Method* object has the main purpose to compute one or more design variables or properties. In this case the attention is not directly addressed towards the assessment of an object behavior but mainly on the computation of one or more specific elements. Such result can however be considered as a specific case of the behavior of a certain element (system, subsystem or components does not matter), and from this viewpoint similar to the scope of *Function Model*, but the conceptual reasons that animate the definition of such two concepts are considered different within the current work. The *Design Method* class reflect the need of dimensioning relations and rules while the *Function Model* class show how an element behaves. Some example can help to clarify such distinction.
The computation for example of the thickness of a structural panel is provided by a rule of thumb based on the height and width of the panel itself (taking into account preliminary dynamic considerations of the panel behavior). In this case such relationship that links some properties of the panel can be model through a *Design Method*. The evaluation of the stresses and deformations of a certain panel when loaded with external forces can be pursued through a *Function Model* that shows how the system under evaluation behaves. Such computation does not directly generate values or data that can be stored within the system model but there is not limitations on such possibility. The computation of a satellite autonomy can potentially be included within both the *Design Method* as well as the *Function Model*. In this example in fact the autonomy can be evaluated as a property and then stored within the system model but at the same time can be seen as something directly representing the behavior of the system. In some cases both definitions can overlap but it is however important to distinguish such objects since they refer to situations conceptually different. An example of *Constraint* class is less "overlapping" with the previous ones and a simple case can be represented by a constraint on mass value for the allowable launch limit.
A design method can be seen as a particular case of a constraint where the relation is always true since the related output are computed directly through the relation itself on the basis of the available input. In particular the relation can be implemented exploiting the capabilities of Modelica language and with an object oriented approach. In this case the output and input are not known before the computation has been executed and the causality of the involved quantities is explicitly resolved with the code run. Other functions or equations can however be considered where the causality of the related quantities is already known before the execution of the code. In this case the input and output of the expression are clearly defined when the relation is created. In both cases the common feature is represented by the fact that the relation is used to obtain one or more quantities on the basis of available values, independently of the fact that the causality of such quantities is known or not. Such aspect is mainly related to the actual implementation of the code and more generally it is also not ensured that the available data allow to compute the desired quantities.
In the case of constraints instead the expression is not resolved to compute one or more input but only to establish if the relation is true or not. In this case the expression is not necessarily true as in the case of a

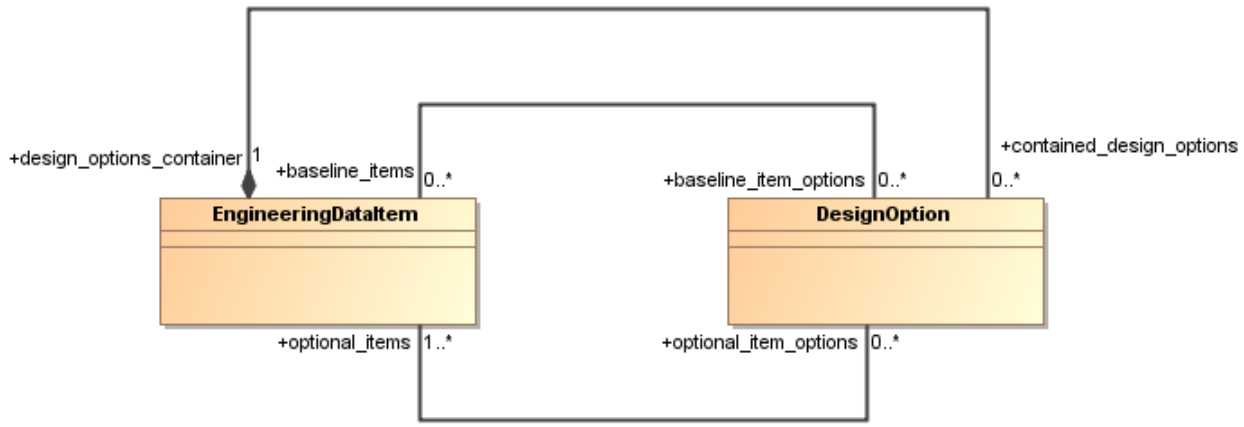

Figure 6.6: Conceptual overview of the meta-model main relationships related to the *Design Option* class.

design method.
The *Design Method* class must not be confused with the definition of *Verification Method* class. The first object is strictly related with the generation of design solutions and provides values with respect to certain properties/objects for example. In the second case instead the *Verification Method* describes basically how are defined the methods that check the correctness of the design choices.

6.3.5 Options and alternatives management

The management of design variables (including with this definition also the options that can be considered during certain design phase and that for example are under evaluation) can be approached in different manners and some of the most current methods have been introduced previously on the state of the art section. In the present work options and alternatives are handled differently with respect to the solutions considered within similar analyses (briefly introduced in the sections above). In particular new classes are defined to exploit the model based approach for such aspect and the related concepts are presented in the following. The proposed solution for the management of the design variables is represented by the definition of the concepts of *Options Group*. This object represents a set of options that are related to the definition of certain design variable. Different options groups are theoretically conceived as independent between each other. Two different options groups are referred to two different and theoretically independent design variables. Potential constraints between two different options groups can be implemented through the definition of constraints class that can be derived from requirements objects. In this manner it is possible to model and capture the design constraints between different design variables but such constraint objects have to be theoretically introduced. The *Options Group* class strictly depends on the definition of another concepts that has been defined as the *Design Option* object. Such element is used to define the set of the specific option/alternative that can be associated to a particular object. A conceptual overview of the related associations are reported in figure 6.6.

In the proposed infrastructure the individual *Design Option* is contained at least within an *Engineering Data Item* since more generally when an option/alternative is defined it always belongs to a father object. At the same time the *Engineering Data Item* can contains zero or an undefined number of design options. A *Design Option* collects one or more optional items that can be represented by *Engineering Data Items*. In the same association an optional *Engineering Data Item* can be linked to zero or more than one *Design Options* since it can basically appears in more than one option/alternative. A specific *Design Option* can contains or not a set of baseline items, represented by *Engineering Data Items*, that represents the nominal configuration for the current baseline of the project. In the first case the class refers to the concept of *alternative* while in the second one it is addressed towards the representation of the *optional items*. In fact when an individual *Design Option* does not have an associated set of baseline items this means that the related objects are not alternative to another set of elements but they are more properly a collection

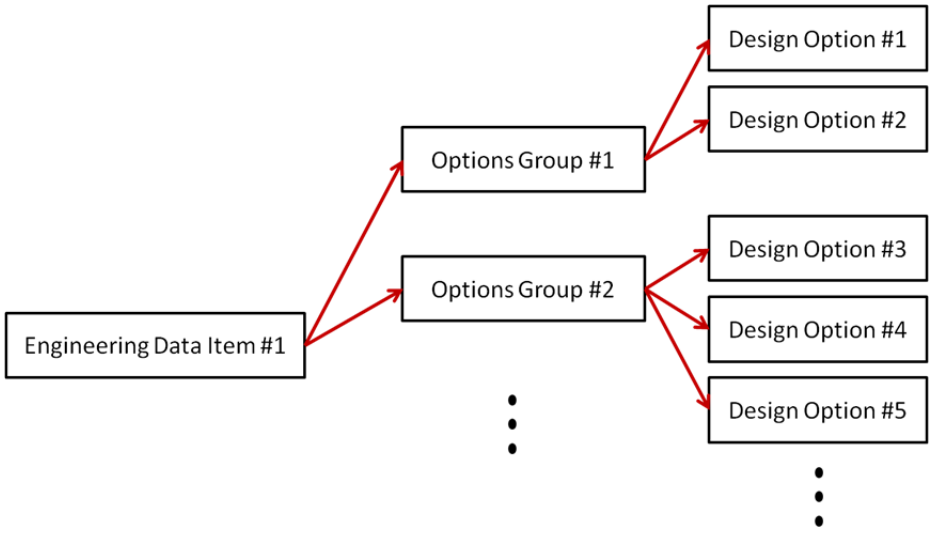

Figure 6.7: Example instantiation of *Engineering Data Item*, *Options Group* and *Design Option* objects.

of optional components. The distinction used in the current work between the "alternative" definition and "option" one is provided in the previous section on taxonomy. The same association allows also to know which *Design Options* are associated to a certain *Engineering Design Item* (as can be seen in the other direction of the association).
The *Options Group* concepts introduced in the initial part of this section gains importance when an evaluation of different design solutions must be considered. In this case the investigation of system performances is based on the correct understanding of which items are correlated and which not. It is possible in fact that some baseline items are common between two different design options and in this case they cannot be considered separately. The presence of such kind of overlap between design elements does not allow to manage independently these objects. It is important then to understand which objects can be managed separately, allowing an effective definition of the overall design space. In particular it is assumed that an *Options Group* represents a collection where the individual elements can be considered as a design variable. In this way when an analysis is performed each *Options Group* contains a set of possible "values" represented by various *Design Options*. From this point of view it can be seen as a useful object to clearly define the set of groups that must be properly processed during an analysis of system capabilities. Other strategies can however be considered for the management of the information available from the proposed model-based infrastructure. The *Options Group* class has been mainly conceived to take into account the possibility to set-up automatic or partially automatic methodologies for the investigation of system performances. Such aspect is however strictly related to the actual implementation of the code and more details will be provided in the following chapter. Summarizing it is possible to say that an *Options Group* identifies a collection of one or more *Design Options* that point to the identical set of baseline items while the same *Engineering Data Item* can contain a collection of *Design Groups*. An example instantiation of such structure is conceptually represented in figure 6.7.

In this way each *Option Group* can be assimilated to a design variable that can assume the "values" represented by the contained *Design Options* (they "virtually" cover the range of the possible solutions with respect to a specific baseline configuration). In the same manner it is also possible to distinguish between a group of alternative (pointing to a baseline set) and groups of options.
The developed definition has been conceived to include not only the optional or alternative items from the physical or topological perspective. The introduced concepts are in fact defined from a more general point of view. In this manner optional or alternative choices can also be considered for other items types, such for example the activities, scenarios or functional items. The main definition is directly built in fact from associations with the basic *Engineering Data Item* (more details about the Engineering Data Item class are provided in the appendices).
The conceptual definitions considered does not limit the creation of optional or alternative elements that contains on their own alternative or optional objects. Such situation is foreseen in the metamodel since the

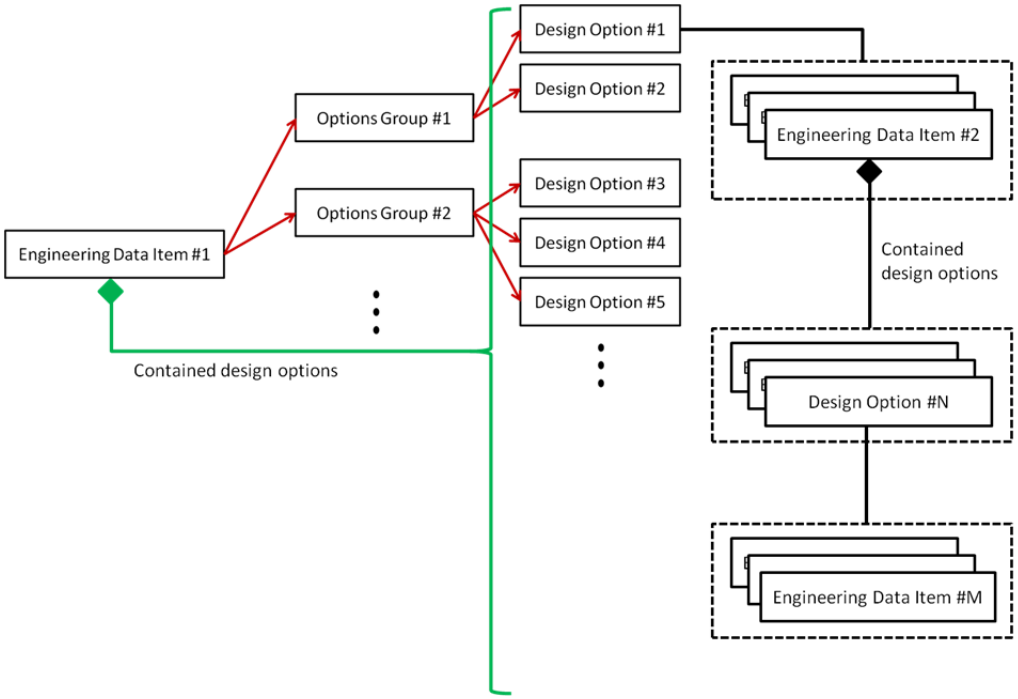

Figure 6.8: Conceptual representation of a scenario representing the definition of optional/alternative objects of other optional/alternative elements.

Engineering Data Items that are defined as the optional/alternative elements for baseline ones can contain *Design Options* since such possibility is not constrained, as can be seen from the conceptual model. In figure 6.8 a simple representation of such situation is reported for the sake of clarity.

Once the representation of the system options and alternatives is defined the management of such information can be approached considering different strategies (their choice depends often on the actual solution that will be implemented). In particular such data are used to properly generate the possible combinations that can be associated with the product itself. Such operation must also be evaluated in two different contexts. In the first case the generation of the combinations can involve the same level of detail for the object, providing the possible combination for the same hierarchical level. In the second one the combinations are instead generated considering also different hierarchical levels, paving the way for the generation of the alternatives tree of a specific product. The related analyses must take into account the possibility that some baseline elements are common among different *Options Group*, highlighting some overlapping of the involved objects. It is important to underline that the *Design Option* class is defined conceptually in the metamodel but the related objects can be directly captured from the information available with *Design Options* and associated *Engineering Data Items*. From this view point they are not necessarily implemented within the platform but are however used to manage the analyses of alternative configurations of the product.

In the case that two *Option Groups* have no common elements in the set of baseline items then the number of the overall combinations between such two groups is represented by the product of the number of alternative contained within each collection. In the case instead the two *Option Groups* have at least one common element from the related baseline items then such two collection are not independent and the number of the overall combinations is represented by the sum of the related alternatives. In the actual implementation of the strategy for the management of product combinations the possible solutions are different. A possible choice is represented by the evaluation of all Option Groups assumed at first as independent design variable and each combination will be investigated to find overlapping objects. If one combination show at least two set of baseline items (the two corresponding to the related *Option Groups* under evaluation) that overlap (with at least only one object), then the overall combination can be highlighted with a warning.

The same representation scheme of alternatives/options can be used to properly generate the alternative tree when there are more than one level of nested *Design Options*. The available information can

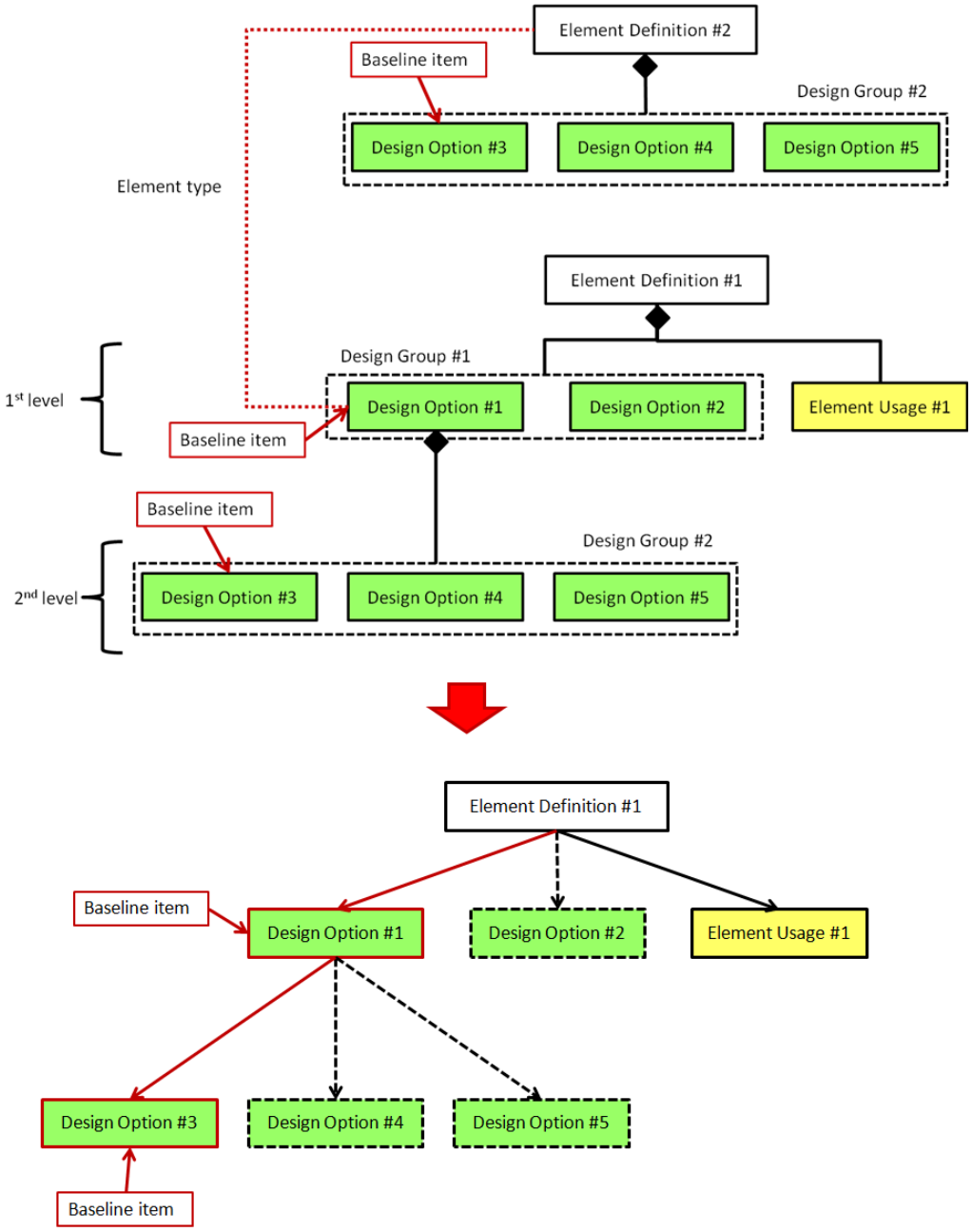

Figure 6.9: Simplified example of the alternatives/options representation on different nested levels.

be used in fact to set up the overall possible solution across the hierarchical decomposition of a product. For example if an *Element Usage* contained within an *Element Definition* represents one of the possible alternatives of a *Design Group* for the *Element Definition* itself, and the same *Element Usage* points to its *Element Definition* (which defines its type) which contains in turn other *Option Groups* (and their related *Design Options*), then such information can be explicitly represented. Starting in fact from the root element it is possible to iterate over the baseline elements and associated alternative solutions to identify the overall nested combinations. Considering the previous example such process is driven by the fact that accessing each *Element Usage* it is possible to clearly identify the contained *Design Options* thanks to the connection with the corresponding *Element Definition* (figure 6.9 briefly represents a simplified example of such connection).

The *Baseline* class has been defined to conceptually store all the items that represent the current status of the system under development. In particular it contains the *Engineering Data Items* that define the system, considering also the nominal items that are included within a *Design Option*. Not all the *Engineering Data Items* are defined to belong to a specific *Baseline* since there is the possibility that a set of objects are contained within the *Project* class (which in turn contains the *Baseline*). In this manner some objects can

Figure 6.10: One of the reference cases considered for properties/options management.

be defined within a *Project* class without necessarily belong to a specific *Baseline*. This approach allows to create items that can be used as alternatives/options for a specific *Design Option* without directly affecting the structure of the baseline itself.

6.3.6 Scenario types

The definition of the approach for the management of design variables has been developed starting from the analysis of a certain set of the possible design cases and actual examples that can be found during the development phase of a project. The conceptual characterization of such methodology has been done concurrently to the investigation of some examples cases with the final aim to assess the correctness of the proposed pattern. Iterating on the proposed solution and example cases has allowed to identify and correct the conceptual model. During this phase the main objective is to assess how the concepts seem to be well suited for capturing the actual design cases (once certain situations have been badly managed then the conceptual model has been modified to take into account such condition). Some of these design cases are reported in the following section, supported by the definition of examples to better represent the related situation.

Figure 6.10 represents the case where an Element Definition A contains two Element Usage. In the first configuration the object B is an Element Usage that comes from an Element Definition that is different from the Element Definition from which is defined the Element Usage D. Also the element C is an Element Usage but it is not characterized by an alternative solution. In this case the object B and D is assumed to belong to the same Option Group because in this representative case it is assumed that both represent a design variable. Actual situation can be represented by a motor-wheel assembly (the Element Definition) that contain a wheel that is defined an it is not a design variable (the Element Usage C). On the other side for this design level the configuration has not been closed for the definition of the element A but two possible solutions for the motor type have to be evaluated. The options for the motor type are represented by the motor B (i.e. the Element Usage B) or motor D (i.e. the Element Usage D), respectively “instantiated” from an Element Definition for the motor of type B and Element Definition for the motor type D.
This example further enhances the fact that one Option Group is uniquely related to one design variable, i.e. one element that must to be evaluated (for example through analyses).
The Concurrent Design Variable might contain several different Element Usages as it is defined (to model the fact that certain virtual element can be associated to different alternative solutions). In this case the Element Usages are not necessarily inherited from the same Element Definition (this depends on the fact that the conceptual model allows for the management of the case where there is the need to model the cardinality of some Element Usages coming from the same Element Definition) but they can be related to different Element Definition theoretically. Their belonging to the same Element Definition is a special cases that can however be encountered within a design process.
An Element Definition characterized by a particular Design Variable that is inherited on the related Element Usages must allow to manage the associated parameters in a independent way. This means that two or more Element Usages that are derived from the same Element Definition can manage the corresponding Design Variable in a independent manner. The values related to this Design Variable can be different between different Element Usages but belonging to the same Design Variable definition (since they are derived for the same Element Definition) they have the same range, mean, variance, etc. They are independent but share the same characteristics (belonging to the same Design Variable definition from the Element Definition).

Figure 6.11: One of the reference cases considered for properties/options management.

Another possible case that can be found during the design phase is resumed in conceptual figure 6.11.

Figure 6.11 refers to the case where the Element Definition A contains a certain parameter that can be still defined. In particular the object B can represents a certain design variable that is characteristic for the Element Definition B. This object can represents for example an Engineering Data Item that can be chosen and its value has not been still definitely taken for this level of development and it is still under evaluation. This Engineering Data Item (also identifiable with the Concurrent Design Parameter from the standard ECSS) is contained within an Option Group element to specify that this object represents a design variable. This assignment allows to manage this element for possible further multidisciplinary analyses since its belonging to trade space is formalized with the Option Group definition.
An actual example for this theoretical case can be represented by the case of a wheel element (the Element Definition A) that contain the design parameter B representing the radius of the wheel itself. In this case for example the range of variance of the wheel itself can be expressed as a continuous range between two ends but also a discrete range can be considered. This depends on the particular design trade space and all this information can be expressed for example within the Option Group element where the Design Parameter is included also if this one exist within the Element Definition itself. In particular the proposed methodology foresees the formal definition of an Option Group that link the design parameter within it but the parameter itself exist before the definition of an Option Group since for example the radius parameter exist for the wheel before also if an Option Group is not defined for the wheel (since for example the wheel itself has fixed radius). The definition of Option Group is something additional to the current definition of the element. The main idea of such an approach is related to the fact that the Option Group tell me that some of the available features of such an element can be changed or tuned. This approach has been implemented since it makes easy the reuse of already defined similar element. The wheel similar to the one defined with a variable radius (i.e. that defined in the example just discussed) can be obtained starting from the just defined one but eliminating the presence of the option group element.
The same Element Definition can potentially have multiple Design Variable objects and not all the combinations of the related alternatives (since all the Design Variables have been conceived as theoretically independent objects) may represent a feasible design configuration or satisfy the requirements. The requirements violations or infeasible solutions can be verified a posteriori (after the wrong combination as been obtained/considered) through an automated process in the case of simple models or by the generation of a more complex analyses model (created from the domain specialist engineers). Both this investigations are however analyses that can be done to verify the correctness of the design solution and must be done a posteriori, also because it is more difficult to proceed in the inverse direction. Generally the design process first defines a possible design solution and then verify through the analyses if that solution is feasible. The last investigation requires a direct involvement of a domain specialist to build the model needed for the analyses. The first type of investigation instead is characterized by a partial generation of the models (that are simpler with respect to those generated in the second case) in automated way through functions acting upon the available design data. Both cases can theoretically be considered as equivalent instruments of analyses to verify the design but in the first case they are supported and generated more quickly (theoretically without the direct presence of a domain specialist). Another case is expressed then in figure 6.12.

An Element Definition A contains an Element Usage B that in turn contains a Current Design Parameter C that can assume different values over a defined range. This actual case can be represented by the motor-wheel assembly where the contained wheel can change its radius. This case however is a particular condition that can be represented as special representation of the previous ones. The Element Usage B

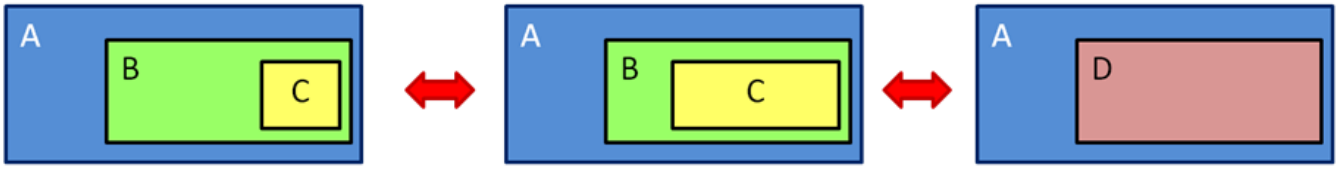

Figure 6.12: One of the reference cases considered for properties/options management.

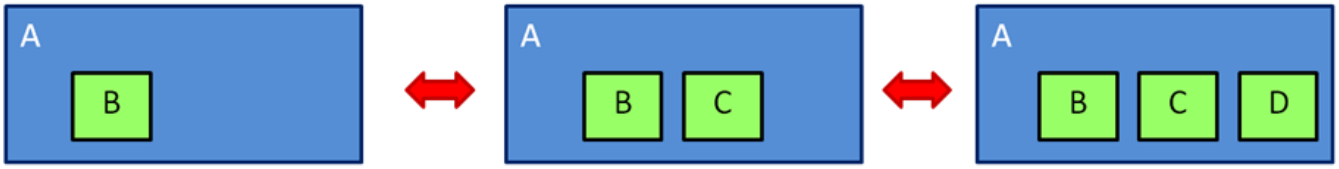

Figure 6.13: One of the reference cases considered for properties/options management.

inherits the design parameter (the radius) since it is an instance of a certain Element Definition where the radius has been defined as a design parameter. This example helps to enhance the fact that the Option Group is defined for an Element Definition and all the Element Usage that are defined from this one all inherit the design variable/variables that are defined in the Element Definition itself. It is necessary to create a new Element Definition if the design variable is not needed for certain elements.

Figure 6.13 represents the case where an Element Definition A contains a certain cardinality of some Element Usages (B, C and D) that belongs to the same Element Definition (they are all instances of the same Element Definition). In this case it is not important the range types for the Element Usages themselves that can be discrete from two ends (one of which can also be the infinite) or represented by an enumeration. An example of such case is a particular design context that can be encountered during the development process but however it can represent very rare situation. An actual example can be represented by the choice in the number of the motor-wheel assembly to be modeled within a certain design (for example the locomotion system as our Element Definition A). In this case it is possible to choose three, four or six motor-wheels assembly (the Element Usages denoted with the B, C and D notation) and all have the same Element Definition.

The design process often is characterized by the fact that this choice on the number of the motor-wheels assembly must be consistent with the number of power cables that have to be considered for the power supply. In this case the consistency is not imposed on this implementation level but can be checked through the use of proper function that can be introduced within the system model and that can be implemented as constraints and obtained starting from the requirements. This helps to underlines the fact that at this level the definition of element and their characteristics are not subject to analysis that are addressed to another level of investigation (the check through functions that investigate for the consistency of the proposed solutions represents however an analyses also if they are simple ones because launched and verified still in automatic without the creation of related models and simulations).

The proposed conceptual definition for the design variable allows to manage particular cases where some elements are strictly constrained about the cardinality of the involved element. This cases can be managed without the definition of constraints between the objects involved. For example the number of implemented motor may represent a design variable and its definition can be considered through the management of a group of Element Usages. This type of situations can be characterized by the fact that each motor implies a corresponding wheel. This constraint not necessarily can be modeled with a proper definition but may be implemented indirectly defining an intermediate layer that is represented by an Element Definition that includes the motor and wheel Element Usages and that it is globally identified as motor-wheel assembly. In this manner each time the motor-wheel assembly "instantiate" a certain object then it includes already the fact that one motor corresponds to one wheel.

The Design Variable object shall allow to define an empty element for the management for example of a special case of cardinality where the absence of Element Usages can also be an optional solution. This case is used to define design situations where it also foreseen the possibility that no Element Usage is needed

Figure 6.14: One of the reference cases considered for properties/options management.

Figure 6.15: One of the reference cases considered for properties/options management.

from trade-off analyses. For example an exploration rover can considers the definition of the number of battery element as design variable. In this case there is the possibility that the solar arrays are enough to provide the required energy supply. Under these conditions the battery Element Usage can then be represented by an empty object because they are not needed from the analyses. In this situation might be useful to have defined the possibility that the design variable can be represented by an empty element. The reference cases that can be considered for the conceptual definition can be represented also by the figure 6.14.

In this case two Element Usages (inherited from two Element Definition) represent the options to other two Element Usages since they can be strictly related and cannot be placed within the Element Definition A without the corresponding Element Usage. This case allows to model the situation where two (or more) Element Usages are correlated between each other. In particular the Element Usage B can be associated to the Element Usage C and equivalently the Element Usage D must be associated with the Element Usage E. An example can be represented (in the motor-wheel context) by the situation where a motor of type B can be associated only to a wheel of type C while a motor of type D can be related only to a motor of type E. This situation can be approached differently. An approach can be represented by the fact that the Element Usages are managed independently and so all the combinations are allowed but external constraint functions (implemented in another context) are executed to evaluate the feasibility of the investigated configuration. In this case all the Element Usages belong to an equivalent number of Design Variables, each one independent from the other. The other approach can be expressed considering two intermediate Element Usages generated from Element Definitions where the corresponding characterization is represented by the containment of the two (or multiple) Element Usages that are interrelated. In this manner it is possible to avoid the definition of an external constraint to model the related condition. With this second solution the constraint is in fact internally bounded with the Element Definition. A conceptual representation of this second solution is represented in the figure 6.15. A special case of the considered situation is however captured with the considered modeling and it is represented by the fact that the two couples are obtained from the same Element Definition.

A complex design situation can be represented by the figure 6.16.

Figure 6.16: One of the reference cases considered for properties/options management.

In this case the two Element Usages are defined starting from the same Element Definition but they are evaluated considering two different values for the same Design Variables inherited from the Element Definition (from which all the considered Element Usages are defined). This design problem is also captured within the defined conceptual model.
One important feature of the proposed modeling approach is represented by the capability to develop both logical and physical models of the system. In this manner is possible to clearly define the difference between the functionality (defined through the definition of logical models) and the hardware/physical objects that allocate such functionalities (defined through the definition of the physical models). The correct definition for the available alternative solutions can help to provide interesting support functionalities. The formal characterization of design options can be directly managed to obtain a clear representation of the system alternative configurations. The information gathered by the correct definition of option class can be used for example to automatically generate pattern as the trade tree which is widely used to provide a useful viewpoint during system development and design [58].

6.3.7 User conceptual model

Considering the specifications available from [65] it is possible to identify the users that are directly involved with the system modeling tool. They are briefly reported and described in the following list.

- **Study Manager**: is the team member that represents the study customer.
- **Team Leader**: is the team member that leads the concurrent design study team.
- **System Engineer**: is the team member that is responsible for system engineering and the overall system aspects.
- **Assistant System Engineer**: is the team member that assists the System Engineer.
- **Domain Expert**: is the team member with particular skills in and knowledge of a specific domain, usually an engineering domain, and responsible for those aspects of a concurrent design study that relate to that domain. How many domains experts participate in a concurrent design study team depends on the specific needs of a particular study.
- **Study Coordinator (Central Authority)**: is generic role indicating the common user aspects of the team leader, system engineer and assistant system engineer.
- **Administrator**: is the user of OCDT that performs administrative tasks to create, adapt and configure user accounts, study areas, IT infrastructure including backup, restore and archiving, etc.
- **Observer**: is a participant not actively involved in the study also if this role is under study and must be confirmed (at the time this work has been written).

The users access architecture has been implemented starting from these theoretical classes and definitions (defined equivalently within the data model). The provided credentials at login page are then used to proper manage the information returned to the current user on the basis of his/her access level. In this way it is possible to customize the data that the user can see, providing different restriction levels about the capability to create, edit or only see certain properties. In the same manner it is also considered the possibility to change the returned perspective (page or framework layouts) on the basis of the discipline-domain the user belong to. This feature has been directly related to the role that certain user covers within a project, allowing for the same user to cover different roles in different project as also various roles for the same project. This capability ensures a flexible management of the people involved in project, providing also the basis for a well-organized collaborative environment. The details about such integration and implementation are introduced in the following sections, where the results about the Ruby on Rails implementation are briefly presented.

The introduced user profiles have been used and slightly modified to develop the roles that are then actually implemented within the proposed framework. The implemented framework has been developed not only on the basis of the previous conceptual model from the standard but also taking into account actual design processes. The roles defined for the proposed approach are briefly introduced in the following list:

- **Administrator**: the users with such role monitor the whole web-based infrastructure and they are mainly involved in the developing and coding activities. Such user has access to all data contained in the database and all functions that can be provided by the application itself. Examples of such role are network or application administrators.

- **Analyst**: this user defines generally a project member that is able to access data (on the basis of the process configuration), make comments and upload certain resources (e.g. in the case some analysis results must be linked). Such user is basically not allowed to create, update or delete engineering items since this capability is associated mainly with the designer role. Such profile has been conceived to model all such project role that generally deal with analysis and simulations but are not directly involved in the actual design process (they can propose architecture or components changes through comments but only the user with item ownership can modify it e.g. the designer). In particular there is no constraint on the fact that the same physical user (the individual person) can cover different role within the same project (designer and analyst) at the same time. In this case on the basis of access credential the user can in fact change the design or edit items with the advantage that the overall process is now formalized and monitored. Examples of such role are thermal analyst or mechanical analyst.

- **Designer**: such user can access the data on the basis of the ownership that he/she possess. In particular such profile can edit the data on which he/she exerts his/her ownership. Basically all the engineering data items that such user defines inherit the ownership from him/her and remains under these conditions until a process owner user modify the ownership of the data that belong to a designer. The actions and operations that such profile can do depend also on the process configuration for a specific project (such operational domain is defined by the process owner). The available data and visible information can also be filtered on the basis of the discipline the designer comes from. Examples of such role are system manager, verification manager or program manager.

- **External Service**: such profile has been defined to model all the actions that can characterize the interaction of the main application with other web-based infrastructure. In this case data can be provided following the paradigm of RESTful interfaces. In particular not all data can be exchanged but only the filtered ones or those designated on the basis of the process configuration. Such profile has been considered to basically exchange data not through rendered views (since the process in this case does not involve another human user but generally a web-service) but directly through data format as JSON or XML. Examples of such role are external tools (with web interfaces) or dedicated web-services (in the case some distributed services are provided on the same network).

- **Process Owner**: such role defines the users that can orchestrate the overall design process of a specific project or program. From this viewpoint the process owner can be seen as the administrator for a specific project. Process Owner can assign tasks, activities and roles within a certain project. She/he can also modify the ownership of a certain object, basically establishing the rules that regulate all the interactions among the various users. Examples of such role are team leader or system manager.

- **Reviewer**: this role is quite similar to the viewer since such profile shows all the features that can be found within it. In particular this user has all the characteristics of the viewer with also an additional capability, represented by the ability to define comments to the related project. The customer can also be included within a project with such role. Examples of such role are customer or internal reviewer.

- **Viewer**: this user has been conceived to define such profiles that cannot create, update or delete items but are only allowed to view some specified sections of data and information. This user has mainly been introduced to manage all the guests that can be involved in a project. In certain cases such profile will be also used to handle the accesses of customers for example.

6.3.8 Quantity, units and properties conceptual model

QUDT/QUDV introduce one of the main concepts that covers a fundamental role for the definition of a correct data exchange process. This terms includes all the relationships that relate the classes for the correct management of the quantities, their units and the associated values. In particular the development of a shared data model about for example the concepts of units of measure, dimensional analysis and system of units can potentially improve the communication between different engineering domains, reducing for example the time required for the right conversions of quantities and values. Interesting initiative regarding the formalization of quantities, units, dimensions and types is represented by the NASA QUDT [59]. The objective of such project is the creation of a semantically enhanced version of standard engineering tables using ontology expressed in RDF/OWL language and terms. QUDT is also conceived to provide web services to support conversions and also dimensional analysis, reducing the error-prone process for data exchange. The main purpose is represented by the creation of a consistent context of share meaning, enhancing the communication capabilities across diverse fields, functions and domains of expertise. Such target requires a standardization of data structures and specification of queries for information retrieval. In this way it is possible to obtain clear improvements for the integration of data and interoperability of processes and tools across the product lifecycle. The main benefits potentially related to this project can briefly be reported in the following list:

- Consistency of data exchanged
- Compatibility of analyses and communication across different fields and domains
- Mitigation of errors and their impact
- Satisfaction of lifecycle development and operational needs
- Structured and web-based access to additional model-based QUDT information, tools and services

The definitions that characterize the overall conceptual model can be briefly identified in the following list. Some of the considered objects have their related element in our proposed framework.

- Quantity Kind (kinds of physical quantities)
  - Base Quantity Kind
  - Derived Quantity Kind
- System of Quantities
- Unit of Measure
- System of Units
  - Base Units
  - Derived Units
  - Coherent Units
- Dimensions
  - Base Dimension

- Dimensional Analysis

A model-driven traceability allows to monitor all the referenced links and their properties. QUDT project has also been conceived considering the compliance with some of the current standards on units and quantities

- CODATA (Committee on Data for Science and Technology)
- BIPM (Bureau International des Poids et Mesures)
- NIST (National Institute of Standards and Technology)
- ISO (International Organization for Standardization)

6.3.9 Product model concept

Another interesting concepts developed within the current conceptual infrastructure is represented by *Product Model*. In particular such concepts has been conceived to model the features that can be directly associated with the models elaborated by the various analysts coming from different disciplines. Such features has been considered inn the current infrastructure, foreseeing the connection with the more advanced phases of a project. The final purpose is represented by the target capability to pave the way for the connection with complex models defined externally with respect to the expected framework. The connection between the external models with the data structure available from the system model is fundamental to ensure a correspondence with the current baseline and information exchanged. In this way it is possible in fact to better monitor the gradual grow of the system and related information. Such structure can be used to create a one to one correspondence between a domain specific model and the system model itself, providing the basis for a more consistent mechanism for the control of product structure. The main idea is represented by the fact that the unique system model can be related to multiple product models that can be potentially mapped to external system models, coming for example from different engineering domains. The domain specific models continues to store the main information related to the related field (since the data associated to them can be quite big), but the contained data can be properly elaborated to populate the associated product model within the system model, filtering for example unnecessary or redundant properties and data. From these explanations it is possible to see how *Product Models* have been basically conceived to provide the link between external complex product models and the system model. *Product Model* is however stored within the system model infrastructure and at leat it includes all the information needed to map the external resources with the common system model. In addition to such basic data (which are directly related to the main role of such concept) other useful data can also be stored, for example when it is important to monitor the property coming from a specific discipline and when such features can be inferred from the external model itself. The concepts expressed so far can also be extended to specific parts or subset of the system since their meaning is not limited to the whole product. The main purpose is represented by the fact that ideally each discipline will have its corresponding product model to map with the system model (supposing that a certain discipline has its own product model).
The information gathered or loaded within the *Product Model* can be used to simulate system behavior with respect to a specific scenarios for example. It is important to underline that such concepts has been conceived to allow for the connection between the models (above all the complex ones) elaborated within the specific discipline and the system model. The data available from the system model are basically used to drive and help the generation of complex models as it has been considered for the main use of the proposed infrastructure. There are however no restrictions on the fact that the domain-specific models can be partially generated from system model, in particular when the design phases are not so detailed. In this context such concept must not be confused with the *Function Model* (as well as the related *Function Definition*, more details are available however from the appendices). *Function Models* are equally used to simulate the response of the product on the basis of the available information mapped with the system model. The main difference is represented by the fact that ideally the product models are produced

within a specific discipline and they already contain all the information needed to be simulated or integrated within a more complex case. Such data can be compared with the data loaded in the system model as well as partially exchanged with it (depending on the main purpose of the integration procedures with system model, "what load which" and "data hierarchy"). On the other hand the generic *Function Model* does not contain the information needed for a simulation. Such element has been conceived to be potentially used with different kind of similar components, as it allows to achieve the response on the basis of the available data. Such data are stored in the system model and are mapped to the function itself which can be theoretically used also in other project. In this case the function model does not contain all the information needed to potentially run a simulation but it must be linked with information available from the model (in the other case the product model is instead linked with an external model which has all the data required). The main idea is to include in the definition of *Function Model* all the codes/simulation items that can be flexible reused on other cases and in other project. As defined the *Function Model* is in fact not directly associable with the product representation of the system as instead is introduced for the *Product Model*. Such distinction allows to clearly separate the models that are generally developed in the advanced phases (linked through the *Product Model*) from the capabilities to simulate some behaviors in the preliminary steps (using more "simple" and flexible models, managed through the *Function Model*). In the same manner there are not limitations on the fact that the *Function Model* can be represented by a complex code sued in the advanced phases of a project. The related choice strictly depends on the available resources as well as the final purposes of the model for a specific case. The main difference remains related to the fact that the *Function Model* does not natively contains all the data needed for a simulation while the *Product Model* can rely on its own information to support a simulation.
The *Function Model* and *Product Model* classes are not totally separated since they can be both used to set up a complex simulation from the information contained within them. In particular the information contained provided by such object can be used to obtain the final Simulation Model.

6.4 Workflow for the proposed approach

The first phase of the present work has involved the conceptual analysis of the workflow and definitions that helped to build the proposed methodology. In particular a clear description and evaluation of the possible alternatives in the context of the considered problem have been first investigated. From these ones a solution that seems to show a better behavior has been considered. This choice regards the conceptual definition of the processes, people and tools involved in the design methodology. Such decisions have been supported by the representation and construction of proper meta-model architecture for a clear organization of the work that have to be done, ensuring that all aspect will not be neglected.
Once the main concepts have been elaborated the following phases focuses on the actual implementation of a prototype infrastructure to assess the approach itself. Different choices can be made among the possible solutions for such step. In this case an approach similar the Agile Development Lifecycle as been followed as much as possible for the realization of the target infrastructure, trying to put into practise the main guidelines of the related philosophy. In this way it was possible to better evaluate the most promising solution for the development of the desired platform.

6.4.1 Agile development lifecycle

An interesting development lifecycle model is represented by the Agile software approach which shows some promising capabilities. It is well widespread in the field of software engineering and development while the application of the same infrastructure in system engineering domain is not formally investigated also if some advantages can be obtained. In particular Agile software development is a set of software development techniques based on iterative and incremental process. The related methods have been conceived to allow requirements and solutions evolve through collaboration of cross functional skills and capabilities provided by different teams. The same methodology can potentially be applied to system engineering discipline, enhancing the integration among the current lifecycle technologies and innovative

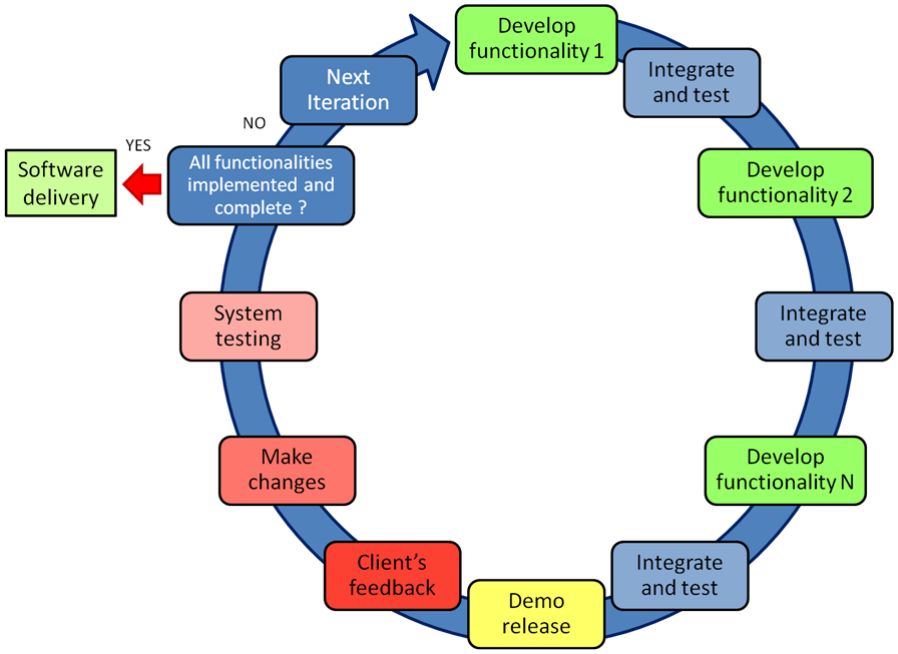

Figure 6.17: Example of Agile development lifecycle applied to software design.

design approaches. Agile methods enhance the promotion of adaptive planning and evolutionary development, encouraging the rapid and flexible response to design changes (a conceptual example is reported in figure 6.17).

Some of the methods define concepts that can be used to improve the current system development process, reducing the possibility of unexpected errors and ensuring the consistence of the product features. All the basic principles are included within the Agile manifesto and are mainly referred to software development process but some of the highlighted concepts can be extended to engineering disciplines. Such shared concepts can be summarized for example in the following list:

- Customer satisfaction by rapid delivery of useful product
- Welcome changing requirements, even late in development
- Sustainable development, able to maintain a constant pace
- Close and daily cooperation between business people and developers
- Projects are built around motivated individuals, who should be trusted
- Continuous attention to technical excellence and good design
- Simplicity is essential
- Self-organizing team
- Regular adaptation to changing circumstances

There are many specific agile development methods which most promote development, teamwork, collaboration and process adaptability throughout the lifecycle of the project. Some of the widespread agile techniques that show interesting features for the design of complex systems are Acceptance Test Driven Development (ATDD), Agile Modeling, Continuous Integration (CI), Feature-driven development (FDD) and Test-driven development (TDD). For example the TDD method provides useful utilities for a clear formalization and definition of the development process, reducing the possibility to neglect some requirements. In this case the implementing activities of the system (the software in particular) starts first from the definition of a structured code listings that are used to check the correctness of the software that have to be

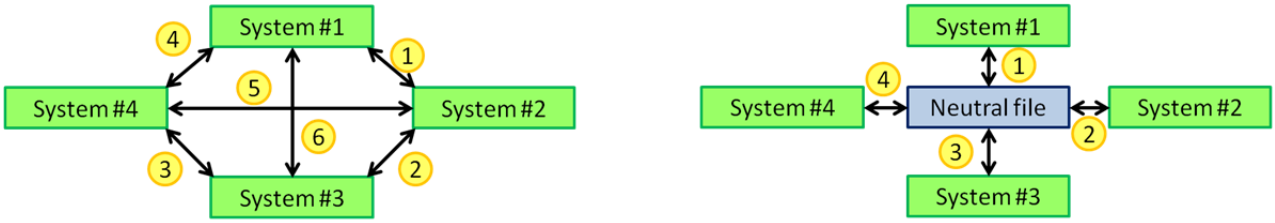

Figure 6.18: Alternative data exchange architectures.

still developed. In this way the requirements and the capabilities that the tool must provide are formalized before the code itself has been developed, focusing on the tests that at the end must be fulfilled. The code software can then be tested once such test infrastructures have been defined, exploiting some automatic or partially automatic execution capabilities. In this way the debugging operations can be partially automated, reducing the workload required to verify the correctness of the already developed code. The same approach of software implementation can potentially be also extended to other engineering development processes, providing some interesting capabilities for the management of simple design activities and verification in the same way.

6.5 Data exchange

Data exchange between different CAD/CAE/CAM systems covers a key role for the right integration of multiple design environments when a collaborative and distributed framework is developed. The main problem is represented by the format consistency across different platforms and different design systems since often the data structures are not shared and some restrictions limit also the access to proprietary database information. The two main exchange methods that can be used are conceptually reported in figure 6.18.

The connection between the various elements is realized through the implementation of proper interface for the exchange of files and data. In A the number of system adapters rapidly increase as the number of involved environments grew up, making such approach difficult to apply when many domains are working on the same project. This situation requires also a well-organized maintenance activity for the management of all the different adapter typologies that are required for the correct working of the whole architecture. In B the number of required adapters is smaller with respect to the previous case since the connection must be properly set only with a central neutral file standard. The efforts required for the maintenance of to the pattern A are not so demanding as in the previous schema since each system has its own adapter with no knowledge about the connection of the other elements. In this way the standard neutral file can be used to exchange information through a straightforward process that reduces consistency problems across different modeling and analysis environments. Standard neutral file is obtained through pre-processing activity starting from native database, transforming than such data to other native database with a post-processing action. Such activities flow allows ideally exchanging information from one system to another and vice versa. This solution must be accomplished automatically as possible, reducing the efforts that the single user spends on data conversion. An example of such data exchange is proposed in figure 6.19

The proposed concepts identify only the actors and resources involved within the related pattern but the same architecture can be actually implemented focusing on different hardware solutions and infrastructures. The main idea is to integrate the concepts related to the second considered approach within a web based infrastructure, exploiting the benefits coming from data exchange on a distributed framework. The most common standard neutral file formats are briefly reported in the following list:

- IGES: such format is fairly widespread for the communication between CAD/CAE/CAM systems and it is supported by international standard organization (ISO).
- DXF: format proposed by Autodesk and mainly used for the exchange of drawing information.

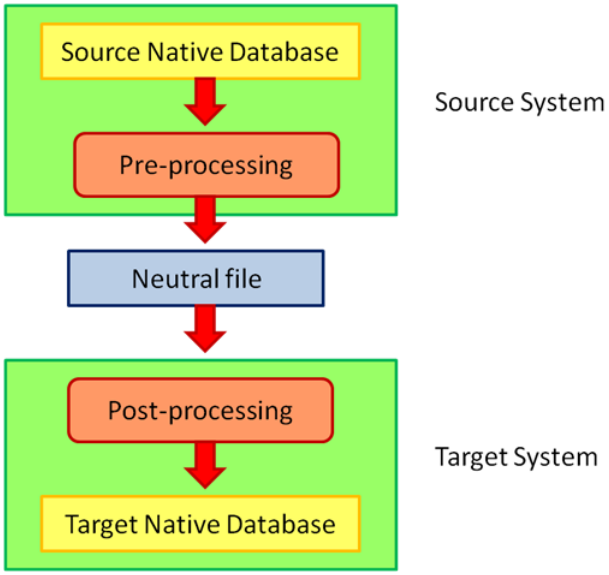

Figure 6.19: Data exchange mechanism.

- STEP: such format is used to store all the data involved in the definition of product life-cycle data: design, analysis, manufacturing, quality assurance, testing and maintenance. Such format is basically knows in the past as Product Design Exchange Specification (PDES).

Nowadays most of used CAD systems are currently moving towards STEP format also if they used to manage IGES data structure in the past. The proposed framework must also consider STEP format as strongly recommended solution for the interface of data coming from other systems. The information contained within the system database can be properly processed through specific adapter to generate the required data. In the same way the system modeling framework can import the information provided as STEP file, paving the way for a two way communication link with other external systems. Data exchange among different domains is often followed by a wide range of issues since an errorless communication between tools, processes and platform is difficult to realize [60]. The identification and introduction of a neutral file format has been mainly animated by the industry requirements with the final aim to gain competitive advantage over the traditional approach. One of the main feature that affect data exchange within also the same company is represented by the consistency check of product lifetimes in information technology. This element is also related to the barriers to communication that can be found during the development of a complex system. Within the same industry a product can span even twenty years from its design until the end of its operational life. During this period the company will probably have replaced its major applications several times, have replaced its systems software at least once and have replaced its hardware several times. Under this condition the maintenance of data is not a negligible problem and data legacy can be ensured through only some investment. The increasing amount of data available in a project often lead to more “meaningless” data as also more information. The management of a large volume of data is characterized by various information costs that come out from different sources:

- Duplication costs or redundant efforts related to the recreation of data
- Maintenance costs related to legacy systems
- Maintenance and acquisition costs for the software used to exchange data among different domains
- Data storage costs
- Costs related to the data access
- Costs due to transcription and translation errors that come out during data exchange
- Costs related to loss of quality

STEP format has been mainly conceived with the final aim to mitigate the just introduced information costs, paving the way for bridging the gaps between different computer systems. Actually the integration among different environments is achieved through different solutions such as manual re-input of data, adoption or introduction of a standard systems and direct translation for example. Neutral format translation and shared databases are other two possible approaches for the management of systems information across different domains and they seem to show the most interesting features.
The opportunities created by STEP format are represented by the capability to freely exchange data between different systems (availability of data), accessibility through standard interfaces, creation/maintenance of shared data environments (reusability of data) and enhancement of quality of data by the use of standard data models and interfaces.

6.5.1 Engineering design model of data exchange

The main part of the work related to the development of the conceptual model used in the analysed methodology is based on the technical memorandum and standard documents provided by the European Cooperation for Space Standardization (as previously introduced). In particular the documents considered are represented above all by Engineering design model data exchange [61] and System modelling and simulation [62]. These documents contain much of the information used to conceptual define the simulation and modelling processes. They are both technical memoranda documents that represent not a normative standards but they include useful guidelines for space systems engineering on a specific subject.
The Engineering design model data exchange introduces the main recommendations for the definition of model based architecture regarding in particular the data exchange process of the early phases of engineering design. The objective is represented by the capability to share information related to the same space systems but referring to different design disciplines. This feature becomes particularly relevant when the modelling of complex system involves often different industrial environments and institutional organizations. In this case becomes very important the definition of a common data exchange process when the collaboration activities are particularly integrated. The final scope is to provide the starting point for the definition of a common environment for the early activities of space system development without neglecting the possibility to extend the same approach also on more advanced and detailed design phases. In this way one of the most important benefits is represented by the availability of a common and shared set of parameters covering all the project lifecycle. The three most relevant points directly related to previously introduced objective are represented by the creation of concurrent design facilities, the effective data exchange across different models and finally the real-time collaboration. The concurrent engineering design processes and the taxonomy definitions used in the current work are mainly contained within the annex available with the ECSS documents and represented in particular by Space Engineering Information Model (SEIM) and Space Engineering Reference Library (SERDL). These resources contain all the information which are directly related to the definition of data model. In particular since these documents refer to a technical memorandum they are intended to evolve into an ECSS standard in the near future. Before the contained data model definitions become effectively an industrial and institutional standards it is necessary that a consensus form must be reached and the related maturity validated.
One of the most important concept related to the integration of a concurrent approach in the design of space system is linked to the Concurrent Design Facility (CDF). Concurrent Design Facility is a multidisciplinary design centre using the concurrent engineering approach for the assessment of potential future ESA missions and it is located at ESA/ESTEC. Another important concept is that related to the Integrated Design Model (IDM) which is implemented and developed within the CDF. The Integrated Design Model represented the federation of information model and engineering tools used to support the design of space mission in the early phases through concurrent approach. Both the introduced elements sustain the integration of concurrent engineering approach towards the deeper integration of the available resources, improving the capacity to satisfy customer needs within a co-operating environment. The support of concurrent design is provided also by the implementation of a database system that it is compliant with all the information and software applications that are linked with the concurrent approach (also named as Open Concurrent Design Server OCDS).

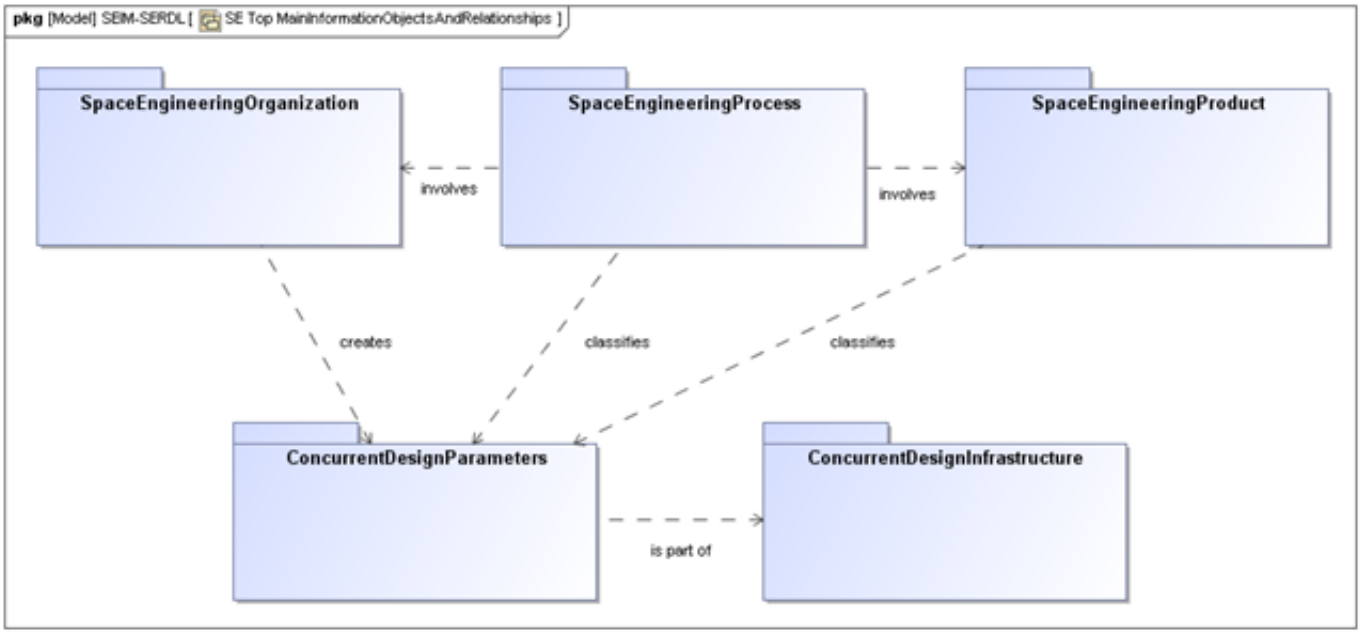

Figure 6.20: Top level view of the SEIM (UML package diagram), [61].

The first developments that characterize the definition of concurrent design approach and that were realized with the CDF started in 1996. The main idea is represented by the need to develop a well shared approach for the definition of the early 0 and A phases of space system life cycle. This concept effectively born for the first time at NASA/JPL within the Team-X facility. Concurrent design approach starts to spread over all the main important aerospace industries where each one has its own internal processes for the definition of space product, each one based on a specific Information and Communication Technology (ICT) support environment. Also if different design environments are involved in the definition of space system, an open neutral data exchange protocol it is fundamental for the successful exchange of data information. One of the main important activities in the definition of the information model and reference data library it is represented by the a clear understanding of which data need to be exchanged. The information exchanged must show the same structure for each study and they come from an information analysis activity realized for the implementation of IDM. There are also other information that characterize the study and that can be described with a common standard approach. For example these items can be represented by system element names and their composition, properties of parts and materials, design input parameters, analysis or simulation result parameters, names of involved disciplines.
The data model structure is based basically on two main parts represented by the following ones:

- Core data model
- Reference data library

The core data model includes all the main definition related to object attributes (defining the class) and the relationships between them. The reference data library represents instead a set of instantiated object that follow the rules, relationships and features contained within the core data model. All these information are needed for the implementation of the proper data exchange and applications interfaces. The information model main partitioning is described with a UML package diagram. The top level view of such SEIM infrastructure is represented in figure 6.20.

This top level view distinguish between:

- Space Engineering Organization
- Space Engineering Process
- Space Engineering Product

- Concurrent Design Parameters
- Concurrent Design Infrastructure

In particular the space engineering process is the element that involves an organization for the definition of the process itself and a product as the object to be designed. The disciplines that constitute the space engineering organization are involved in the creation of the design parameters. These are then related to a specified phase of the process (as the related link represents) and describe a well-defined component/part of the product. The design parameters are stored and handled within a design infrastructure as the OCDS. Each package contains the concepts that allow to specify a particular study and they are classified with different classes. There is one object class types for each package. The concepts under the same package refer to the same classifiers. In particular:

- Organization package refers to organization class
- Process package refers to process class
- Product package refers to object class
- Design parameters refers to data class
- Design infrastructure refers to facility class

System Engineering is currently characterized by one of the most challenging evolution in design processes of complex products. This discipline cover a fundamental position in the right design of system project and the related performances are strongly influenced by the correct management of the involved engineering domains. Briefly speaking the reported diagram define also the relationships between the considered classes. The engineering process involves the engineering organization and engineering product. At the same time the engineering product and engineering process classify the design parameters. These ones are created by the engineering organization. Finally the design parameters are part of the design infrastructure. The main concepts that characterize the object types classes can be summarized in the following list, considering the various packages.
Engineering Organization is characterized by:

- Concurrent design role
- Participant
- Discipline
- Organization

Engineering Process is characterized by:

- Concurrent design activity
- Concurrent design session
- Concurrent design activity phase
- Life cycle phase
- Iteration
- Snapshot

Engineering Product

- Option
- Mode
- System
- Element
- Mission phase
- Property
- Equipment
- Subsystem
- Instrument
- Sub-equipment

Design Parameters is characterized by:

- Concurrent design parameter
- Parameter group
- Quantity kind
- Parameter value
- Parameter unit
- Parameter reference

Design Infrastructure is characterized by:

- Study discipline workspace
- Study IDM
- IDM template
- Study report
- OCDS server
- Documentation and tools

All the objects defined within a particular space system design can be classified following the previously introduced object types. The object types just introduced can be associated through the use of a well-defined relationships as reported in figure 6.21.

The correct definition and modeling of system need a well-established representation of the main relationships between the elements that characterize the system itself. In particular it is important to clearly represent the system decomposition from a hierarchical point of view in order to highlight possible inconsistencies between the objects that make up the overall product structure. Each element that define a complex system must conceptually have its own role and related definition, reducing the possibility to model a system in the wrong manner.

System decomposition can be represented in figure 6.22 where the relationships between the element involved in the hierarchical definition of system characteristics are introduced.

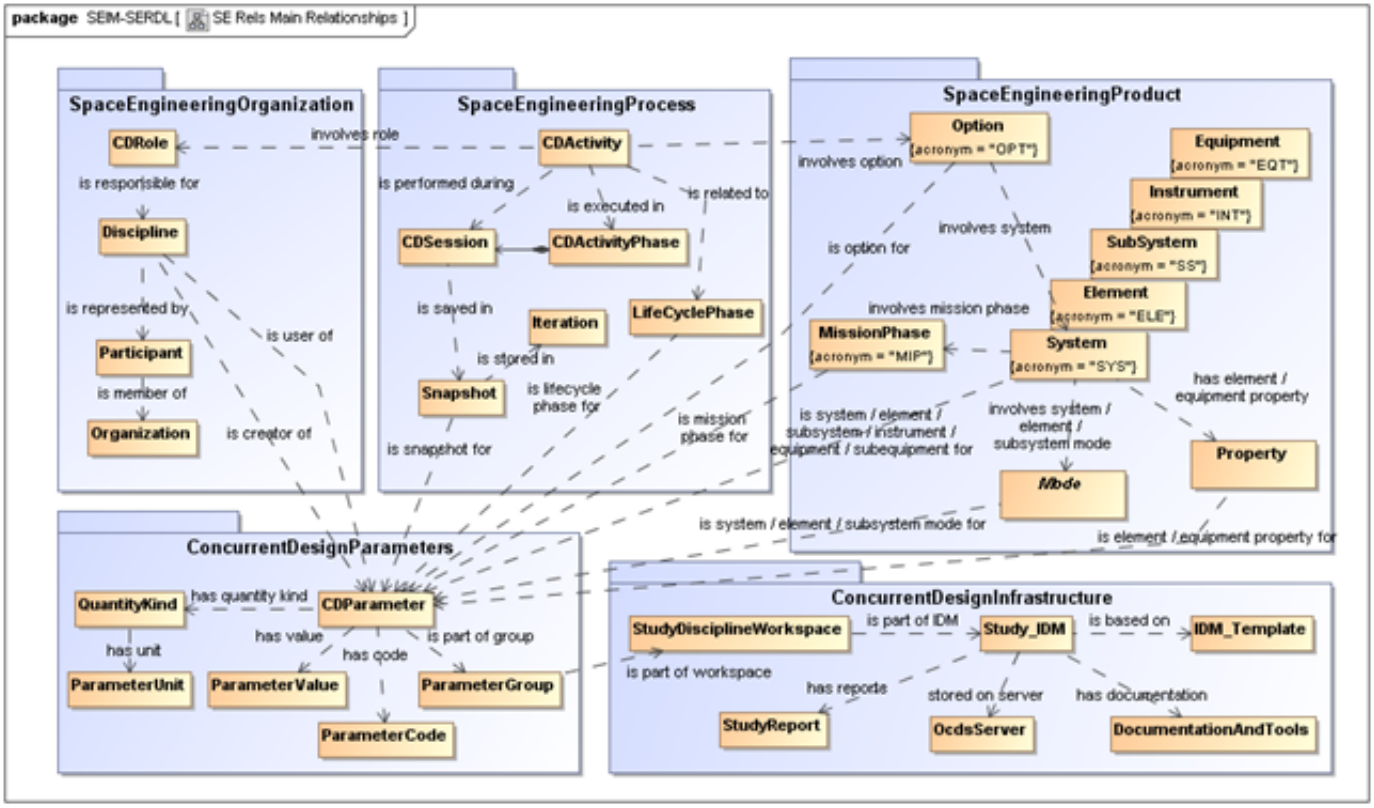

Figure 6.21: SEIM main information object types and relationships (informal UML class diagram), [61].

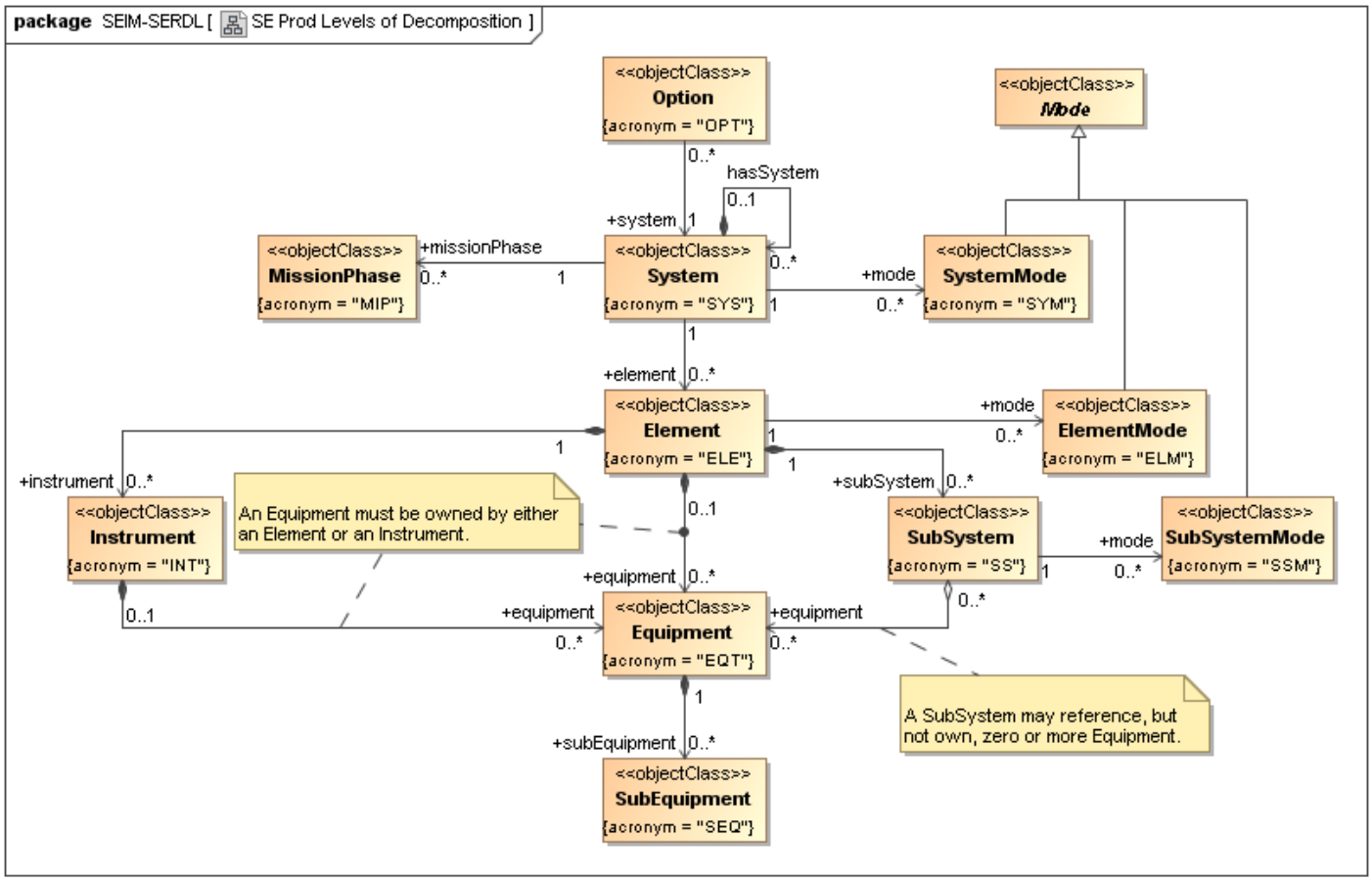

Figure 6.22: SEIM system decomposition and associated modes (UML class diagram), [61].

In particular the reported representation refers to the decomposition of system structure well suited for an adequate description above all of the phases 0 and A. The considered classes are taxonomically defined in the following and the current work mainly refers to these definitions for the research topics concerned within this study.
Option term represents a certain design option considered in the study. In particular there is the possibility to have more than one option for the same system. The word System identifies the top level system. This object can be decomposed in a series of child systems (has enhanced by the hasSystem relationship). System contains Elements while the single Element may contain a certain number of Equipments. This subdivision takes part moving to lower levels through the system hierarchical decomposition. There is also the possibility that at this level the Elements may contain a certain number of Instruments. This latter object can also contain in turn a number of Equipments. The same Equipment instance object can be owned only by a certain Element or Instrument. It does not belong to both these class of objects. The class Element can also contain a certain number of Sub-systems. The word Sub-system identifies the logical grouping of Equipment while the Equipment and Instruments that refer to the Element class represent a physical decomposition. At the same time the term Sub-system can refer to a certain number of Equipments but this relationships is not a physical owning association, meaning that the Sub-system does not physically owns this object (this relationship refers to the logical viewpoint). The Equipment class may contain a group of Sub-equipment. The relationships that this structure introduces concerns mainly the definition of the space product package. The class Systems, Element and Sub-system may be related to different levels of operational modes.
One of the main important features that characterize the evolution of project design is represented by the concurrent design parameters and the related information exchange between the different disciplines. The term Parameter identifies the class that bring the information related to a particular design condition or baseline. The data related to the Parameters are fundamental for the characterization of the actual system performances and structure. They are fundamental for the right representation of conceptual design during the different phases that define the project evolution. In SEIM representation the main role is covered by the *CDParameter* where CD stands for concurrent design. Another important class is represented by the *ParameterGroup* which allows to create a set of parameters that can be managed in a similar way. The class *QuantityKind* specifies what kind of quantity the parameter refer to. A quantity kind may be represented for example by length, mass, time, etc… Quantity kind is related to the Unit class which refer to the measurement unit used for expression of design parameter (m, kg, sec for example). The characteristics of *CDParameter* may also be defined through the definition of the *ParameterValueType* class. This class allows to relate the considered parameter to the kind of computer data used to express its value. A *CDParameter* is also related to *ParameterCode* class which is a coded representation for the referred parameter. Parameter can also be associated to a constant value (in the case the related numerical value is assigned without any particular link with other processes or element) or to a computed object. In this case there is for example a mathematical expression from which depend the value of the related parameter. A *CDParameter* is linked to a precise owner that is identified with a particular Discipline that has a key-role in the management of the considered parameter. The reference data library also known as System Engineering Reference Data Library (SERDL) introduces a set of pre-defined instances on the basis of the object types that come from the SEIM data model. This library includes elements that are common in the definition of concurrent design study and can be regularly extended and updated on the basis of the additional data objects. This library includes for example instances that refer to the Structures, Thermal or Propulsion disciplines. Concurrent design roles as Team leader, System engineer or Domain engineer are also defined. These objects are both obtained from the classes contained within the Engineering Organization package. The SERDL also includes the instances that are related to the classes defined within the Engineering Process package. In this case instances of life cycle phase is used to describe the Phase A, Phase B, etc.. of space system development. Concurrent activity phase class is instead used to instantiate process as Study, Study Session or Study Reporting for example.
One of the main important feature is related to the definition of the parameter code name related to a precise element of system component during a particular design phase. Naming convention follows the data structure hierarchy and generally it depends on the origin of the parameter or on the result that as-

sociated to a particular computation that involves the parameter itself. It is important at this level that the source and the context of parameter value computation are well defined and agreed for the right concurrently exchange of information. The parameter code used within the SERDL library follows the hierarchical decomposition on different level. The same order is traced in the definition of parameter code convention. The following list introduces the decomposition level used for the definition of parameter code name, starting from the higher one the lower. Some level can enhances nested branch that depend on the nature of the considered parameter and its related position within the system design concept and context.

- Option
  - System
    * Mission Phase
      · System Mode

System Mode can be further nested.

- System Mode
  - Element
    * Element Mode
    * Element Property
    * Subsystem
      · Subsystem Mode
    * Instrument
      · Equipment
    * Equipment
      · Equipment Property
      · Sub-Equipment

All the reported level can refer to a particular concurrent design parameter and each element that is used in the definition of the related coded name is generally followed by an integer number that uniquely identify the which of the available objects are under consideration. For example the project may contains two different options at higher level and the distinction between which is under evaluation the parameter code is built starting from OPT1 or OPT2. In particular the name convention introduces the following acronym list for readability reasons.

- OPT Option
- SYS System
- MIP Mission Phase
- SYM System Mode
- ELE Element
- ELM Element Mode
- ELP Element Property
- SS Sub-System
- SSM Sub-System Mode
- INT Instrument

- EQT Equipment
- EQP Equipment Property
- SEQ Sub-Equipment

This acronyms are then concatenated with the integer values related to the specific alternative under evaluation (as previously introduced) and finally ended with the Parameter Reference name (denoted with the PR name acronym). The various level and their identification number are joined using underscore character. Unit names and their symbols are compliant with the International System of Units (SI) as defined in the standard [ISO/IEC80000]. For compatibility reasons between different modeling framework and software application the unit symbols are encoded in ASCII character strings (more details are available in [63]).

6.6 Collaboration mechanisms

A fundamental aspect that affect the application of the proposed approach is represented by the capability that can be provided in the context of a collaborative environment. The proposed infrastructure must take into account all the features that directly influence the implementation of an environment and all such elements must be properly conceived to avoid unexpected situations. The overall framework must in fact also approach the design problem considering the interaction among different users with different roles on the same project. The capability to edit elements properties must be based on specific rules that prevent for example the possibility that two users modify the same object. Some of the conceptual considerations elaborated in this section have been deeply considered during the implementation of the infrastructure since the such topic strictly depends on the actual implementation. The approaches that can be actually used are affected by the codes, languages and database functions investigated. For these reasons the concepts briefly described in this section are only defined on conceptual level but are however introduced to underline the importance with respect to the overall methodology.
The editing activities of a certain object are based on the mechanism of ownership. In particular an element can be created by a specific user as *Element Definition* (more details about such object are provided in the appendices) but the ownership of such object can then be associated to other users (directly on permission/action of the administrator or process owner). Only the users that possess the ownership of such element can change the element itself but this do not imply that they can change the type of the contained elements for example. They can change only the properties belonging the same level of the current object they are working on. The type of the contained elements can in fact belong to other users who possess the related ownerships. Two users cannot work on the same object at the same level of definition (i.e. they cannot modify the same *Element Definition* properties). If a user is working on a specific object and he/she is trying to include some contained elements (i.e. introducing the *Element Usages* to define the architecture) it is important to highlight the elements which type (i.e. the corresponding *Element Definition*) is currently under editing. In this way it is aware of which elements can potentially change during his/her editing activities of the father element. Such interactions can however rise up only when the father and child definition are done at the same time. In the cases where the definitions involve more than two level of hierarchy detail (i.e. when the child of the child is modified concurrently with the father) such highlighting is not necessarily required. More generally the concurrent modification of object related by one level of hierarchy detail can also be done without problems if the editing activities that characterize the child do not affect the its external interfaces. In this case the child objects can be seen as black boxes and if the external interfaces remain the same (from the father point of view only such aspect matters since are the interfaces that are directly involved in the definition of the architecture of the contained elements) than the user can however proceeds to the connection of such contained objects. Such approach cannot however be applied in the same manner to other design activities that involve properties that are defined in the child but are used in the father (for example a design method that computes father property on the quantities contained in the children). In these cases a proper mechanism must show to the user which

elements are currently under modification.
The elimination of properties that affect other elements (for example properties that are used in the father object to compute or define other quantities) must be taken into account to avoid dangerous consequences on data losses. Such situations can be approached considering proper developed alert functions that show the affected elements and consequences that follow from such deletion.

Chapter 7

Analysis, Design and Implementation

The current chapter focuses on the analysis and design of the proposed infrastructure. In particular the conceptual infrastructure built in the previous sections is used to lead the actual implementation of the framework. In this phase the analysis of the theoretical structure is done to ensure the feasibility of the developed concepts with respect to the available technologies. The main aim of such activity is addressed towards the identification of the possible lacks regarding the proposed method as well as the modifications that must be considered to actually implement the overall architecture. The next step is represented by the design of the framework section regarding the integration of the functionalities directly related to the advanced phases of the project. In this case the main purpose is the definition of the target capabilities that the tool will provide once implemented, ensuring the correct identification of the needs and requirements. The fulfillment of such objectives is pursued in the next phases of implementation when the actual integration and implementation of the code is realized.

7.1 Methodology followed

The main research objective is to evaluate, further develop, and apply PSEs for MDA using existing high-fidelity solutions methods. In the first part of the activity a deep and well-documented analysis of the current research projects in such field will be done. Main features, benefits and drawbacks of the investigated methodologies will be listed. This study will be the starting point for the development of methodology and theoretical infrastructure that will be used for the definition of the PSE framework. A clear understanding of the target capabilities and functionalities that will be implemented plays a key-role for the right development of the proposed problem solving environment. The data structure of the problem solving framework will be defined also considering the current research initiatives and available standardization guidelines. In particular such references can be found in the formalization work that was developed both from NASA and ESA research initiatives. Different system modeling methodologies has been developed and tested in these contexts, addressing large efforts towards the evaluation of model based approaches in the design processes of complex systems. A model based approach has shown interesting advantages in the management of a wide range of data and resources. The same methodology can also be used for the definition of a problem solving environment. Such an approach would reduce the consistency-problem between the information exchanged, introducing at the same time a more effective environment to face engineering problem. A PSE tool can also reduce the time spent by scientists or engineers on processes as data exchanges or model transformations with the possibility to devote more energy and resources on activities such as modeling and simulation.

The correct set-up of engineering problems and the related solving methods becomes particularly difficult when different domain specific tools are integrated. The capability to drive the user through the definition of a multidisciplinary problem is one of the most interesting and useful features that can be provided by a PSE tool.

The main part of the current research activity will be addressed towards the realization of a PSE framework after the preliminary phase of data structure and architecture definition. Different implementation solutions can be considered on the basis of the chosen architecture. An interesting and particularly chal-

lenging option is represented by the development of a web-oriented infrastructure. Such alternative will be evaluated as the leading choice among all the potential ones. A lot of European agencies and Aerospace companies have started to develop web-oriented frameworks for the system-level modeling activities. Such environments are basically derived from model-based system engineering (MBSE) paradigm with the final purpose to improve the collaboration among various disciplines across all the design process, from the early phases to the more detailed ones. The same philosophy can also be considered for the development of a PSE tool. A web-oriented architecture has highlighted interesting benefits in a collaborative environment as experienced by some research initiatives ([65]). The same approach can potentially be adopted for the implementation of multidisciplinary environments where the main focus is the correct definition of engineering problems set-up. Such approach will be considered as the first choice and the following activities will be based on this architecture unless an equally effective solution comes out from the previous analysis.

The following phase will be the evaluation of the current web-development platforms to select the more suited one for the desired objective. Open-source projects offer some interesting alternatives that can be chosen to support the development process as a wide range of scripting languages that can be used to integrate the required functionalities. Ruby on Rails project represents one of the most widespread solutions for the creation of web applications and tools. The same functionalities can be found in Django framework which is based on Python languages. A similar approach can be identified with Node.js platform which provides useful instruments for web services development. Another option can be represented by the PHP: Hypertext Language (PHP) which is an open-source scripting language widely used for web applications programming. All these alternatives show benefits and drawbacks with respect to each other and a preliminary evaluation must be done on the basis of the target capabilities that the PSE tool should provide.

Ruby on Rails platform shows useful utilities as also a wide range of already validated and supported libraries. The validation mechanisms, the object-oriented philosophy and the well-defined relational database infrastructures make this solution one of the most suitable.

In this phase the conceptual data structure will be further discussed and developed with the main aim to include the key elements of a distributed environment. The used definitions will follow some of those developed in [66] with further changes to take into account the PSE integration. A web-based tool will be developed starting from this conceptual infrastructure with special attention to the utilities that will be used for the multidisciplinary problem definition. The knowledge, skills and abilities of the people involved in such process are fundamental to make the right decisions and the proposed model philosophy will help to reach this objective.

The MBSE methodology that will be used in this work derives from the context of software engineering and it is based on the connection of simulation models with a central and shared system model [67]. The application of PSEs for MDA will be further investigated considering also the current representations of models based on available formalization initiatives. An example of such research activities can be identified with the definition of the conceptual data models (meta-models) derived from the current ECSS (European Cooperation for Space Standardization) data structure. A system engineering conceptual data model will be further developed to ensure compatibility with current standardization efforts. This Model Driven Engineering (MDE) approach will be pursued with an object-oriented view of reality, using the definition of classes and related relationships as standardization mean in the interaction of different models [66]. The main relations that will be considered in this approach are represented by instantiation (an object is an instance of a class) and inheritance (a class specializing another class). The construction of meta-models will require the introduction of a set of rules for the formal definition of an object-oriented framework. The model taxonomy of this work will potentially follow the definitions introduced by Eisenmann, Miro and De Koning [11].

A web application prototype will be first developed for conceptual data structure validation and then further implementations are planned to assess the management capability of a multidisciplinary problem. All the classes belonging to the above mentioned conceptual data structure will be used to implement a model-view-controller pattern, allowing the instantiation and management of the related objects. Conceptual classes and related associations (compositions, type relationships, etc.) will be formalized and defined. The meta-model will be organized in packages that reflect the user point of view and that will be used to

build the user interface. Each conceptual class will be transformed in a RoR model, and the model instantiations (objects) will represent system items, whose persistence is guaranteed by a relational database generated by dedicated code (migration). This will allow the installation of the web application on the top of any relational database whose adapter is available in RoR (almost all the most used nowadays). Also typical CRUD (Create, Read, Update, and Delete) actions will be defined using basic RoR generators and these features will also support the problem definition, driving the choice to the well suited solving techniques. The resources management will be implemented following the Representational State Transfer (REST) style available with RoR platform. REST "philosophy" defines a series of constraints imposed upon the interaction between system components. One of the benefits of REST is that it scales relatively well for big systems, like a wide network, encouraging the use of stable, long-lived identifiers (URIs). The REST style in Rails is represented by methods to define resources in the routing system, designed to create a particular style, order and logic on your controllers and, at the end, on the way the application interfaces with external world. REST support enhances the advantages of a database-backed application and closely used with CRUD actions will widely improve the organization of the framework architecture and the exploiting of the integrated resources within such environment.
The back-end and a skeleton for the Graphic User Interface (GUI) will be then generated exploiting the infrastructures provided by RoR. The hard-coded part will be mainly the user interface (views), based mainly on web pages, with forms, tables, generated SVGs (Scalable Vector Graphics) and potentially also 3D models navigation. Even this part will be built to be compliant with multiple meta-model classes and robust to their changes.
Dedicated interfaces could also be developed for compatibility with external models (i.e. generic xml models, such as ECSS-E-TM-10-23 and ECSS-E-TM-10-25 drafted models, CAD-derived properties files and Modelica code for example). This activity will strictly depend on the target capabilities that will be proposed in the first part of the work since it will lead to a secondary objective with respect to the main one. The use of a web application for the multidiscipline problem definition will allow the concurrent access of different users, able to navigate according to defined filters and to access to simulation services from remote. A cross-platform web application will help to coordinate the available computing resources, making a better use of the high-fidelity methods and tools already developed.
The developed platform will have the advantage to provide synchronous functionalities, showing for example live updates, reducing the error-prone process related to data and information consistency. The same web service will also provide the possibility to eventually manage asynchronous processes. This need will be further investigated in the first part of the conceptual work and it will depend on the features that shall be implemented. Whilst data consistency is guaranteed by validation methods defined in the model elements (derived by the meta-model), the major issues will be related to generated data items and problems settings.
The generation of standard formats referring to the information collected within the PSE will allow the communication with external tools, exporting for example the data required for a specific simulation. The same mechanism will be used to import information in the proposed framework, exchanging data between different environments and improving the collaboration among the engineering disciplines involved in a project. The implemented framework will provide for example exporting/importing functionalities through XML or JASON extension on a web based platform. Existing libraries can be integrated to manage other standard file formats with little modification of the main application code.
The Domain Specific Languages (DSLs) currently developed for space domain are conceived mainly to be scalable, enhancing the capability to be used at different industrial levels by different disciplines in a collaborative manner [68]. This approach can also be used to properly define a library of categories enabling a well-established semantics of data items. Some research initiatives are addressed towards the evaluation of such methodologies. ESA OCDT for example is one of such activities (as already introduced) and its main application area is represented by the preliminary development phases. Another interesting ESA study is represented by MARVELS which has been defined to analyze the application of such MBSE methodologies within the verification process. In this case the main focus is represented by the investigation of scalability issues with respect to customer-supplier relationships through also different verification levels and stages. Another challenging application field is represented by the formalization about the validation of require-

ments in various engineering domains, involving mechanical and thermal environments as also avionics and functional ones.
In this work the framework has been developed paying particular attention on the capability to monitor the development process through a different layers of role-based, discipline-based and ownership-based rules that regulated the relationships between the involved resources. In this manner it is possible to clearly define the actions the single user can do and data available, providing also a custom perspective on the basis of the role and domain associated with the user access credentials for example. Such infrastructure can potentially be integrated with other tools, paving the way for the definition of not a single and multi-domain application but of a group of interoperable applications. Such objective is pursued through the definition of common concepts and related interfaces.
Model persistence related to the current work has not been achieved through XMI standard as other similar framework implement but other solutions have been considered. XMI standard is in fact not properly suited for the proposed approach while web technology seems to be a promising solution for the management complex models in a collaborative way. A well-formalized development can also help to reduce the efforts required for the maintenance and upgrading of the proposed infrastructure thanks also to large communities that work and continuously enhance such technologies. Web applications can be generated from conceptual models through Ruby on Rails and related generators, ensuring the persistence of information on databases where data structure is based on code migration of conceptual classes. Object-oriented scripting languages (as Ruby or Python) are also employed to interrogate databases and other resources with the final aim to get the information needed.
Other research initiatives as EC/FP-7 Use-it-Wisely project are assessing the capability to use simulations (partially generated from system model) as an additional service to be provided to potential customers, to elicit and to validate the requirements in product-services infrastructure. One of the main important target of all such research initiatives is also represented by the demonstration of the cost-effectiveness of each one proposed MBSE methodology.
Moreover the current work is also interested in the definition of some user-friendly methods (mainly web-based) to map or transform concepts, libraries and formats, allowing non-specialist users (i.e. people with low or no programming background) to have benefit from transformations and interfacing for example.

7.2 Proposed framework

The development of features directly related to the advanced phases of a complex system and designed in the context of a model-based philosophy has been conceived alongside a wider infrastructure. The evaluation of a management strategy for options and alternatives as well as the integration of MDO techniques within a model-based architecture belong in fact to the broader context of DEVICE project.

7.2.1 Introduction on DEVICE infrastructure

DEVICE stands for Distributed Environment for Virtual Integrated Collaborative Engineering and the main aim of this research activity is represented by the investigation of model based methodologies supported by the development of web-based technologies. In particular this project has been developed on internal research activity at Thales Alenia Space Italia (COSE Centre) and the present work reports the results related to the interfacing of multidisciplinary design optimization techniques within such a framework.
DEVICE includes different research initiatives that are all addressed to the evaluation of the benefits of a model-based design and analysis process. One of the main interesting feature of such architecture is represented by the web-based tool for the management of system model. This study has been characterized by different phases with the final objective to investigate the feasibility of integration between web-based services and MDO solving procedures. In particular detailed descriptions of the activities that have done during this survey are presented in the following sections.
DEVICE project includes different research topics as previously introduced but all are conceived to be integrated in a common framework where all the instruments communicate between each other. The main

work behind this activity is represented by the development of a meta-model pattern for the proper definition of data structure. This allows to better organize the data exchange process, ensuring that the formats available are well established between the various applications. The main project considers also the definition of interface elements for the connection with Virtual Reality environment with the target to include such a technology in the design process. The advantages of such integration is directly identifiable both in the prototyping process and in the decision making activities. The support provided can strongly reduce the time related to the evaluation of design configurations alternatives that are often difficult to manage and investigate in the context of team working sessions.
DEVICE includes moreover other adapters and tools that have the main scope to improve the actual design approach, introducing progressively model-based applications in the context of the traditional methodology.
One of the current prototypes developed within DEVICE infrastructure is represented by the Web Editor. The Web Editor is a web-based environment developed by COSE Centre in Thales Alenia Space. A common meta-model was used as reference for modelling activities in the workpackage of different research projects, taking into account the current state-of-the-art in international standardization activities (e.g. ECSS-E-TM-10-23A), as well as Thales Alenia Space Italia and Politecnico di Torino current studies. Using a conceptual meta-model gives the possibility to standardize how information is exchanged at System Engineering level, enabling the actuation of a MBSE architecture and providing a semantic definition of data. Data-exchange formats currently used (standards and proprietary) are not replaced, but it is given the possibility to organize them with a link to the specific element, behaviour or activity related to the system under definition.
The Web Editor meta-model allows the generation of different system model views for the end-user, and also during the design and definition process. These views are:

- Topological Design
- Functional Design
- Operational Design
- Requirement Definition
- Verification Definition
- AIT (Assembly Integration and Test activities) and Operational Activities Definition
- Other more specific (the Discipline-Analysis views)

7.3 Analysis

The analysis of the possible solutions related to the integration of model-based methodologies in the advanced phases of a project is fundamental to identify the most promising choice. In this case analysis refers to the clear understanding of the features to be considered within the framework before the next phases of design and implementation proceed. In particular it is important to rightly choice the solution that is better suited for the final aim of support the engineering activities in the advanced phases of a project.

The framework under development is conceived also to test the possible support capability in the context of collaborative environment, considering the potential benefits that can be introduced from a sizing/design perspective. The same environment can be defined with the objective to final provide interesting functionality for the management of system alternative configurations or design variables. All these elements can be modeled and properly integrated in the meta-model under development, paving the way to a formal definition for the design and sizing procedures/processes.

7.3.1 Scenarios definition and functional analysis

Other initiatives related to the evaluation of MBSE methodology within the design process can be found in the literature. For example [64] provides interesting results about the analysis of different model-based approaches for the management of the information in the early phases of product development, starting from the requirements analysis to the functional decomposition and allocation. In particular the main objective of this study is represented by a survey on different approaches for the design phases related to functional decomposition and function allocation.
Model-base design approach can be defined starting with different approaches that manage the information available from the customer needs. In particular in the study [64] two methods are analysed. They are quite similar but enhance different ways to understand and decompose system requirements. The study available shows both this ways and underline the characteristics of the related features and capabilities. The two presented design approaches are conceptually adopted for a clear understanding of stakeholders requirements and relationships between them and the system functionality. The first one is identified as Usage-driven design approach while the second one is called Feature-driven design approach.
In the case of Usage-driven design approach the first phase is represented by the definition of the scenarios where the system of interest is used in an operational context. Secondly the following action is represented by the consolidation of top-level functions from scenarios and allocation of them to the proper elements. Then the top-level functions are elaborated with internal scenarios that identify sub-functions (proceeding through a decomposition activity iterating from the top-level elements). Finally the identified sub-functions are allocated to logical sub-systems.
The Feature-driven design approach starts instead from the specification of top-level functions (identifying the features directly related to the functions of system of interest) and the following allocation to system of interest. Once this operation has ended the functions are decomposed into sub-functions for the system of interest. Finally the sub-functions classified are then allocated to logical sub-systems. The last phase is common to both the approaches.
The final result is in both case the allocation of sub-functions to logical sub-systems but starting from different initial set of information. This conceptual workflow of activities proceed iteratively during the project development, involving different levels of details.
Generally a usage-driven approach ensures that functional requirements are traced directly to the user's functional requirements, allowing for a better consistency with the user's needs. The feature-driven approach is the traditionally used one and in this case the features, functions and capabilities are identified for a system by domain specialist or/and engineers consulting the end-users. From this viewpoint the features identify those functions that the system have to perform while the usage can be considered as the integration of system features applied in a particular context to satisfy the user's requirements. This survey shows as the analysed methodologies can help to clearly define the system functionality within a model-based context. Starting from the scenarios that describe the system desired behaviour (for different usage conditions) the top-level functions are consolidated and then the lower level sub-functions are allocated once the system of interest components of the higher level have been identified and modelled. The usage-driven methodology starts from the scenarios identified as black-box structures, associated in turn to a particular system components or subsystem that can however accomplish to other functions. The internal sub-functions are then identified and the allocated sub-systems are then studied as white-box structures. The same process is iteratively applied to the design of . A conceptual image is reported in figure 7.1 and is referred to [64].

The same study has also investigated the application of parametric diagrams for the evaluation of Measure of Effectiveness (MOE) and Measure of Performance (MOP) indexes. This concept provides useful instrument to perform a well consistent definition of the quantities involved in the evaluation of system performances, reducing the misunderstandings and error-prone process of data exchange since a unified representation of the information is implemented. The architectural design follows the definition available from the ISO-152288 standard. System complexity is managed through the definition of three main level of abstraction, starting from functional architecture (directly related to the customer requirements), then passing through the logical architecture and finally considering the physical architecture. The last level

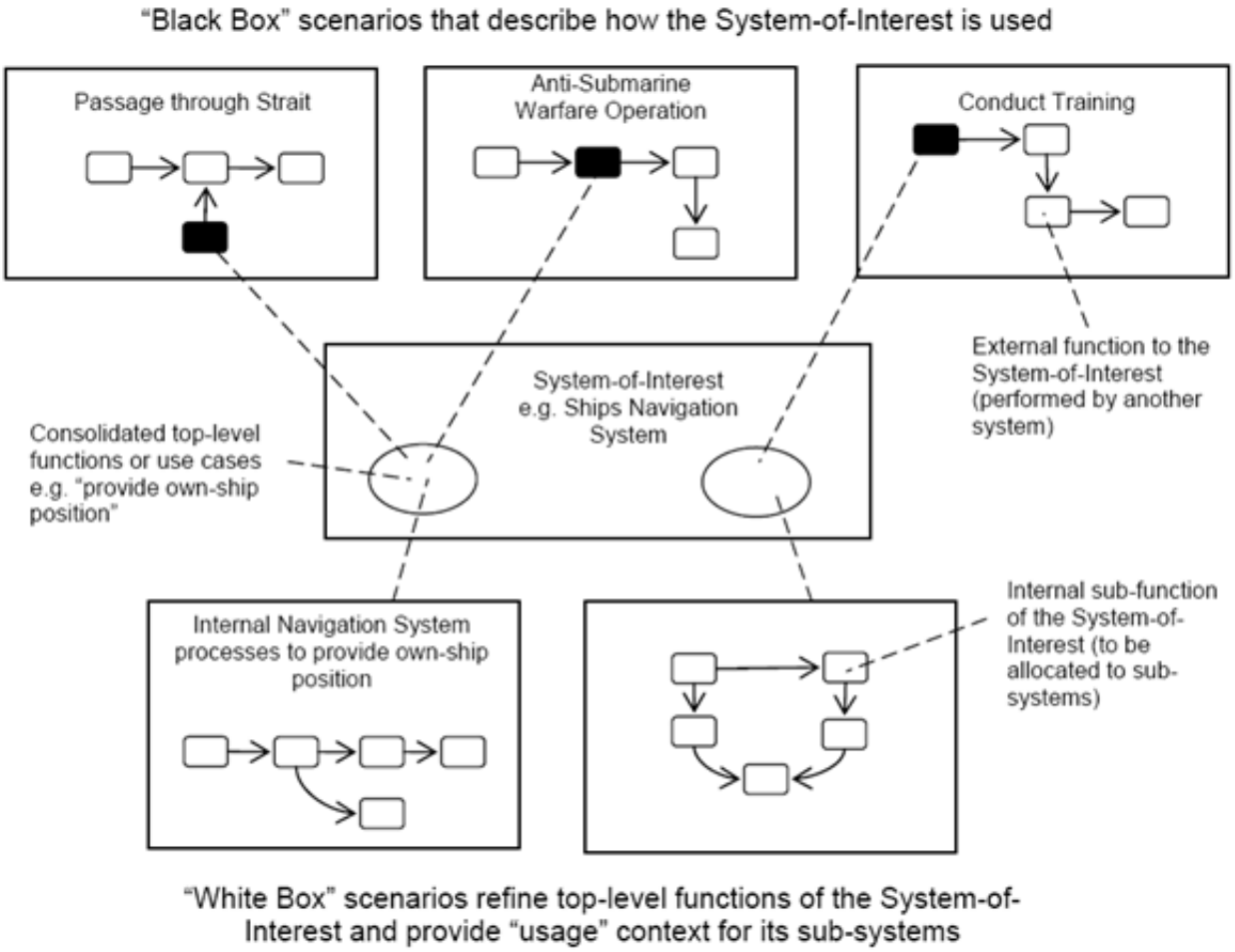

Figure 7.1: How scenarios define and process the system under evaluation [64].

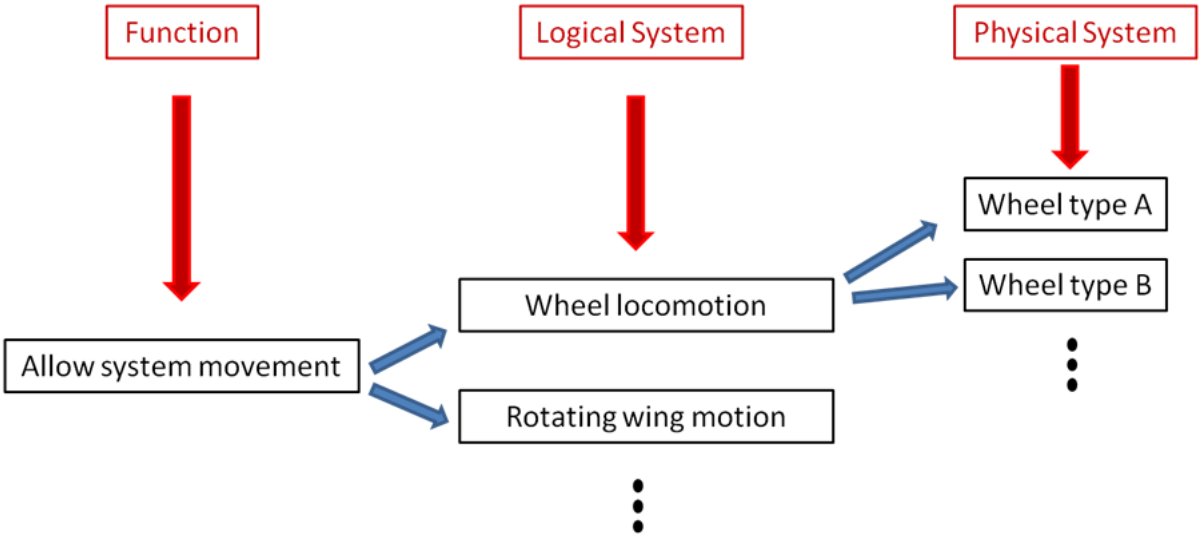

Figure 7.2: Example of conceptual allocation between functions and physical systems.

(physical layer) is closer to real-world level of details. The different levels of abstraction are introduced to help the management of system complexity. As the development process proceeds system definition becomes less abstract and concurrently the volume of design data increase.
The distinction between these tree levels of abstraction is particularly important. From the same function different logical system can be conceived and each one can in turn be allocated to various physical system. An example of this conceptual definition is reported in figure 7.2.

7.3.2 Assumptions and development considerations

The development process of a system can often be characterized by assumptions related to certain quantities as properties or operational conditions for example. This approach is particularly highlighted during the early design phases but also in the more detailed ones it is possible to introduce some assumptions when for example particular elements or entities are not exactly known. Assumptions can often affect design choices since they can also be related to customer needs and requirements. Poor design information from customer require the introduction of some hypotheses about system alternatives and configuration solutions. This feature must be properly managed also from conceptual point of view to model the possible scenarios. In the developed data-model this concepts has been considered and Design Variable class allows to manage also this situation. This object allows to model for example a certain range of values (such as continuous range or an enumeration) related to certain properties within the modeling environment. In this manner it is possible to take into account for certain quantities not exactly known, providing useful elements to formalize assumptions. This approach allows an enhanced tracing of the design hypotheses made during the development, monitoring in a more consistent way the decision making process.
The Design Variable and the options management concepts were proposed with the current model-based methodology and their characteristics have been assessed through the implementation of additional framework functionalities. The features directly related to these objects can be developed in different manners considering various alternative approaches. An example of the possible integration is represented in figure 7.3.

In this case a specific object is directly related with two group of elements. The first one contains all the optional objects that can replace the linked element while the second one includes all the design variables belonging to it. A series of optional objects requires also the proper definition of all the interfaces that allow the connection with the other elements that can be present in the same parent context (container element). At this level it is important to understand how wide is the design freedom since different difficulties can arise from the chosen approach. For example new interfaces is the only additional elements that must be defined if the alternative objects have the same interface port as the nominal one (the element they are linked with). In this case the various specific interfaces (basically the object connecting two ports)

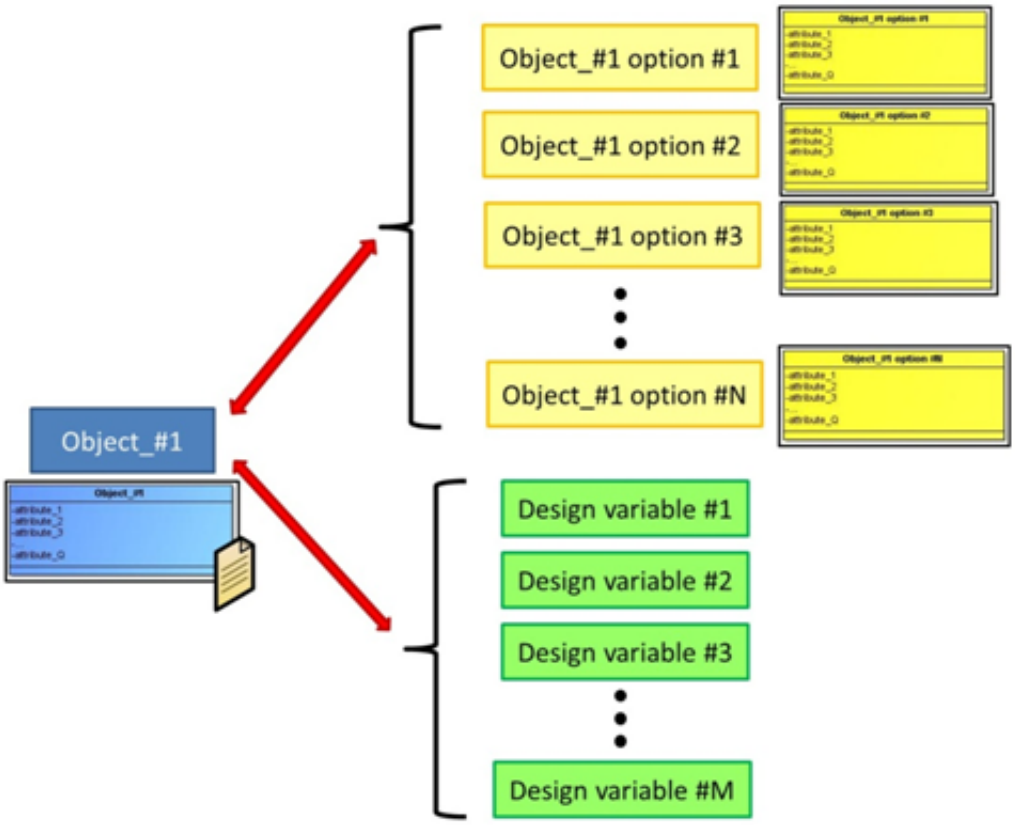

Figure 7.3: Options management and design variables integration.

can be managed through the use of something like active design layers that show which connections are used on the basis of selected nominal element. The management of alternative objects that do not have the same number of interface ports with respect to the nominal one can become dangerously complex to handle. In this case the relationships between the involved objects can be difficult to trace and properly control. The option object defined in the current work has been conceived to model design alternatives (in particular also black-box entities) referring to the same topological architecture for the element that contains the nominal one (and all the objects related with it). The definition of wrapping object can be used in the case the main topological architecture (related to the parent element) is different from that associated to the nominal one. Slight local variations from the nominal architecture can be managed through another level of definition that in the end must ensure the same number and characteristics of interface ports. A simple example of these concepts is reported in figure 7.4. From this viewpoint a new Element Definition must be defined from scratch when a specific alternative object requires a complete change of the overall architecture. The choice between these two alternative approaches strictly depends on the design problem and its specific needs. Some features can help the creation of the new Element Definition which is required when the choice of some contained objects imply the introduction of a new architecture. In this case some of the object properties can be duplicated from the previous defined element and then all the changes can be done from a starting point which has already some instantiated characteristics. The same considerations can be done when the Element Definition requires the introduction or removing for example of external interface ports due to the definition of slightly similar internal architectures. Such situation coincides again with the redefinition of a new Element Definition.
All the previous considerations can also be valid in the case the management of alternative solutions involve the Interface Usage objects. In particular the data structure foresees also the possibility to associate a set of alternative Element Usages to the nominal Interface Usage. This scenario has been conceived to model the situations where the same architecture and the same Element Usage objects can be connected through different Interface Usages but valid anyway. The interface ports are the same since the Element Usages are the same and in this situation a set of different connecting interfaces with the compatible interface ends could exist. The traceability between the various Element Usages which are alternatives to a

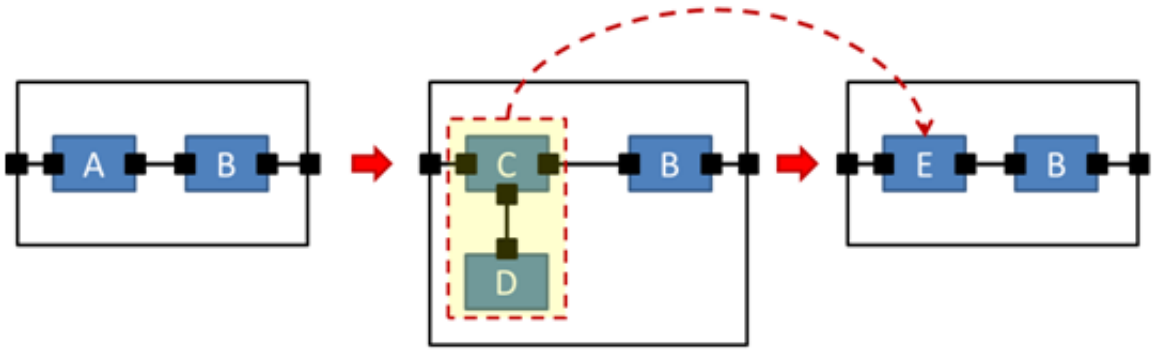

Figure 7.4: Management example of slight different topological architecture.

nominal one and the related Interface Usages (that are require for the connection with the other elements already present in the nominal architecture) is ensured by the fact that the single Interface Usage is bound to a certain Element Usage. It will depend on the characteristics of this latter (if nominal or not) if the related Interface Usage belongs or not to the current design layout.

The main idea is to approach design problem with an object oriented methodology as much as possible. As previously introduced if an architecture is particularly different from an alternative one then a new Element Definition must be defined, starting for example from quite similar object. In this way the implementation time required to build the new element is however reduced. When the architecture is not so different also changing various alternatives for the contained elements then the same object and related architecture can be modeled using alternative Element Usages.

The management of options is provided though the implementation of a dedicated section of Element Definition workspace. The same section is used for the definition of both the alternative groups and the objects declared as design variables. As previously introduced such integration allow to clearly separate the modeling environment for the current design baseline from the variables that can be used to set multidisciplinary surveys. There are no limitations on the possibility to map the object classified as design variables with some components properties but it is important to keep both these concepts on two separate layers. In this way it is however possible to realize multidisciplinary analyses but without the risk to erroneously merge the data available. The design activities can run through the introduction of data and parameters from users while multi-domains surveys provide independent functionalities concurrently to the on-going development process.

The alternatives to a contained Element Usage of a certain Element Definition are managed through the conceptual infrastructure previously introduced. Such approach allows to theoretically define a layered representation for the possible options that must be evaluated at a particular stage of the development phase. The alternative objects can be easily traced to their connections with the other contained elements for the same Element Definition. In the same way the interfaces belonging to the nominal configuration (through the nominal Element Usages) can be distinguished from those that are instead related to the alternative Element Usages. A conceptual example for identical architectures with different alternative solutions is reported in figure 7.5.

The interfaces that connect the alternative Element Usage with the other objects contained within the same Element Definition can be easily identified since they are directly mapped with the Element Usages itself. The same approach allows also the management of multiple nominal Element Usages (with their alternatives) under the same Element Definition. The connections between the objects belonging to the same architecture can be uniquely identified from the interfaced elements. Such an infrastructure paves the way for a consistent monitoring of the alternatives configuration tree, providing useful and well-formalized information from which all the possible combinations for the considered architecture can be investigated.

The elements belonging to the class CDVariables and referring to design variables are defined through a user form that allows to clearly provide ranges, nominal values and numeric types, following the needs of a particular design specification for example. These information are stored and are then used in the case

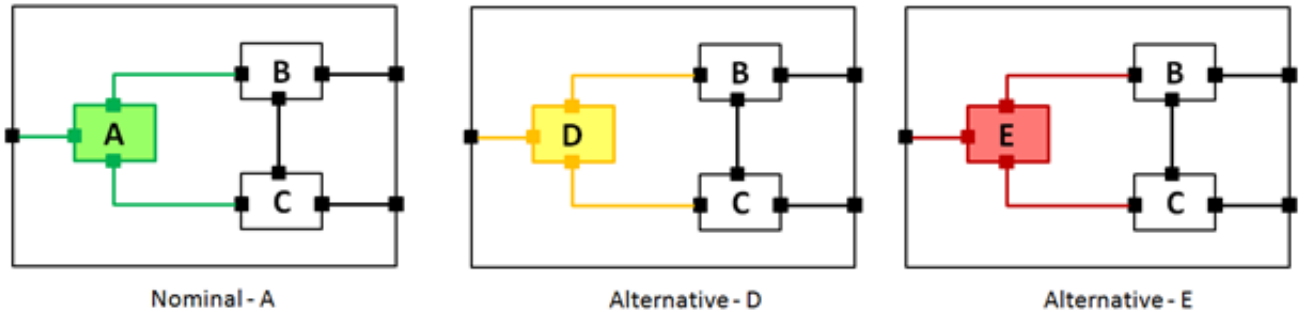

Figure 7.5: Conceptual overview of the layered representation for alternative element Usages and their connections.

the set-up of a multidisciplinary analysis requires such variables as input data. Such design variables are then directly associated to an Element Definition as also the Element Usages that are linked to optional elements. These relationships allows to identify the objects that can be used for a multidisciplinary design analysis on the basis of the selected Element Definition.
The previous approach can be considered in the case the alternative object follows basically the same physical architecture. In particular such situation can be faced through the definition of Element Usages with the same structure of the nominal one but the management of other solution with quite different architecture become difficult to handle. In particular a different conceptual approach will be considered and the previous situation can be equivalently represented as particular case of such new strategy. The previous considerations continue to be valid while the following ones are used at the implementation level of the framework. The main concept used to model such situations is represented by Design Option class. This element is conceived to be associated with a set of items that represent the nominal ones, referring directly to the baseline of the project. The set of items refers to a group of elements that cannot be considered separately for the current modeling level and it can include interfaces, multiple different components or other equivalent objects. At the same time the Design Option class is also related to a collection of items that represent the alternative to the nominal set. In this manner at each design option is possible to link both the nominal group of items and the alternative one. It is however possible to identify the alternatives that refer to the same nominal set since such information can be built from the Design Option class. The comparison among all Design Option objects that belong to a certain item allows to clearly isolate different groups of set of items that are linked to the same nominal elements. In this way the alternative design domains can be defined on the basis of such groups.
Design Option class can also be used to equivalently map a group of items belonging to an optional set (the previous distinction between optional item and alternative item is still valid).
This approach allows the definition of a more generic structure for the representation of system options and alternatives, enabling the capability to capture more complex situations. It can however be used to manage more simple scenarios where there is only a little change in component typology but the architecture remains the same.
A conceptual example of the possible design problems that can be managed with such strategy is reported in the following figure 7.6. It shows how a set of nominal items (i.e. belonging to the current baseline of the project) can be associated to a group of different design options which singularly represent alternative solutions. Items belonging to the same level of detail can be represented without the introduction of a dummy element that should have ideally included the related objects within it.

Complex scenarios can be considered and taken into account since the modeling approach is more generic and enable the designer to monitor design status much more clearly. In this way it is possible to check the consistency of the possible design options that have been modeled, implementing for example specific routines to monitor interfaces correctness. In particular such approach allows ideally to keep under control possible overlapping design options sets since the current strategy do not limit the possibility to consider partially overlapping design solutions. It is possible that some items that are related to the nominal design are also partially linked with another design option together with other objects. Such situation

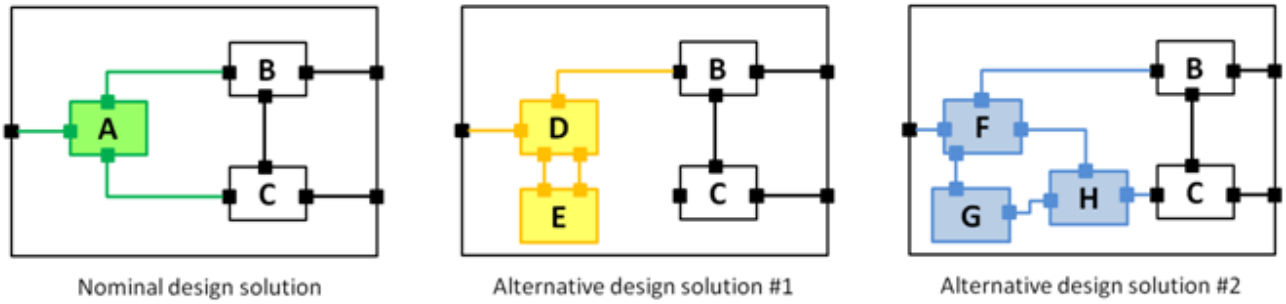

Figure 7.6: Conceptual example related to the management of alternative design solutions.

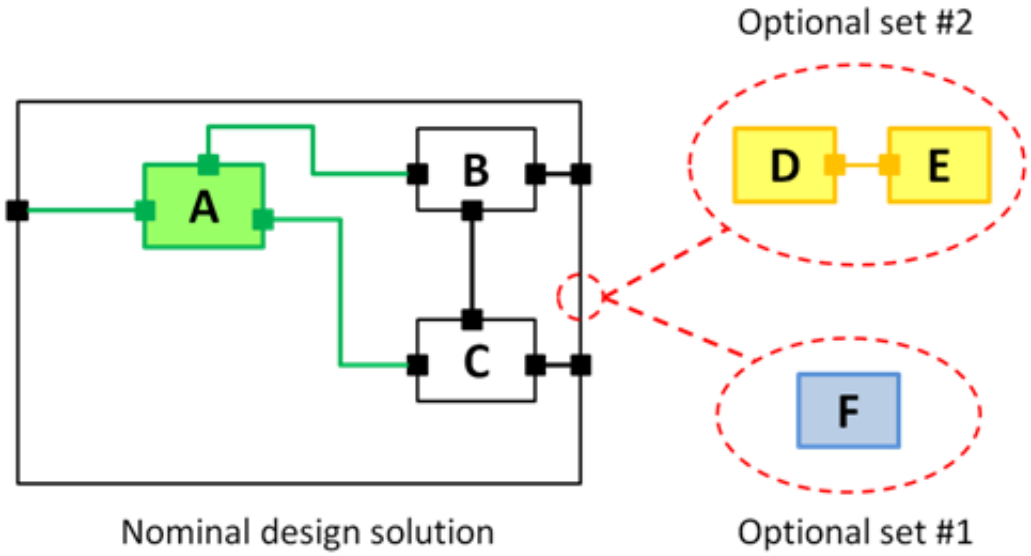

Figure 7.7: Conceptual example related to the management of optional design items.

can occur also when the parent object contains design options sets that are quite separated between each other but that contain a little number of common items. Such method lays however the foundation for the generation and management of alternatives tree. In figure 7.7 is instead represented a simple case where the design option object is used to handle a group of optional set of items. They are basically independent between each others as can be derived from the definition previously introduced. In this manner they can be present at the same topology level and the correctness of their simultaneous presence must be checked through other modeling utilities.

The figure above shows the relationship between a parent object and the contained optional set of items. In particular the design solutions that result from the possible combinations are reported in figure 7.8 for the sake of clarity.

Another important aspect related to the management of design alternatives and options is represented by the capability to formalize the possible presence of nested structures. Some optional collections of items can in fact contain in turn alternative elements that must be properly considered to avoid consistency problems.

7.4 Design and implementation

7.4.1 Introduction

This section provides more details about the actual design and implementation of the proposed infrastructure. A brief overview of the considered languages as well as the development platforms is first introduced, describing the main features of some valid alternatives. After this initial investigation the ac-

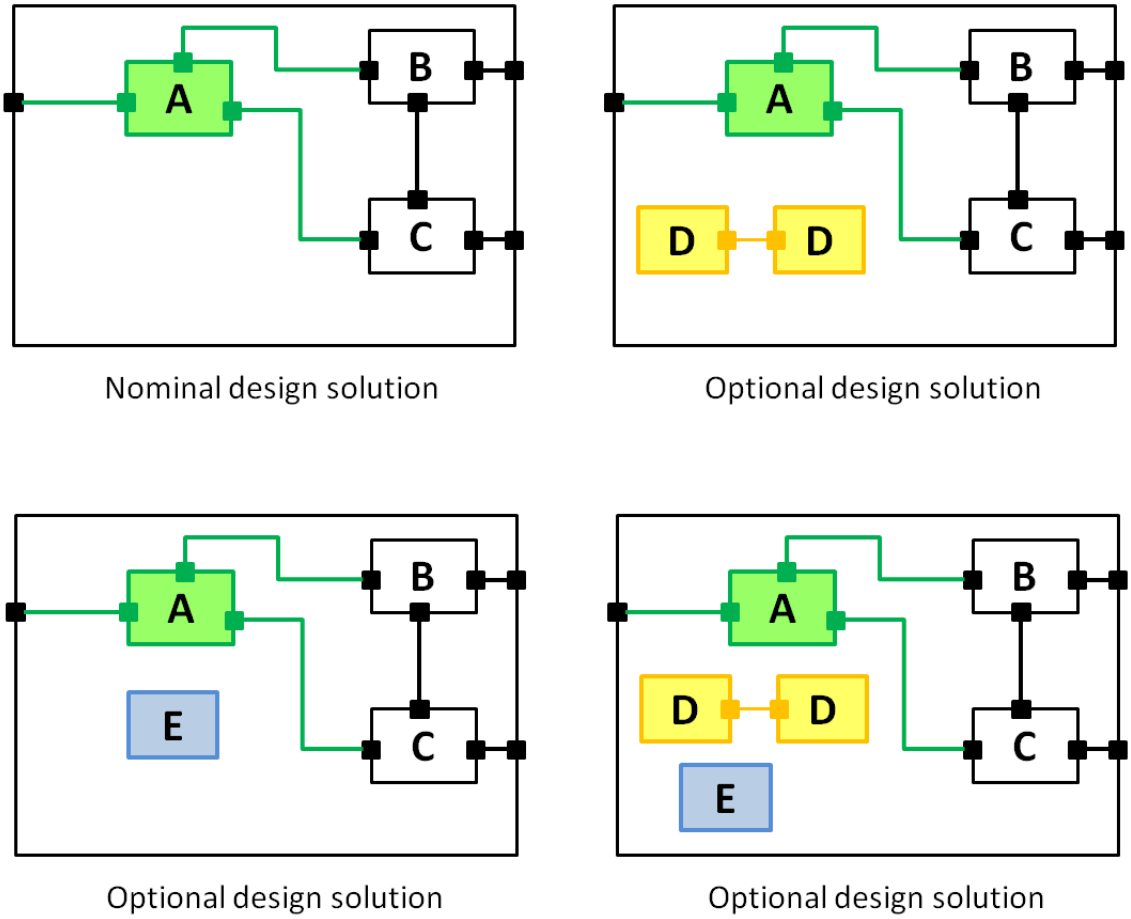

Figure 7.8: Combination of the optional design solutions that come out from the previous example.

tual implementation is done through those assessed as the most suitable. In particular the choice has been motivated by different factors but other approach and technologies can also be considered. In the current work the final aim is in fact mainly addressed towards the evaluation and assessment about the validity of the core meta-model. The enhancements of the developed infrastructure can however be pursued and investigated once the base concepts and their relations are demonstrated.
The implementation of the web-based application has been done with Ruby on Rails environment. This choice has been animated by the wide availability of libraries and functionalities that support the integration and development of this kind of platform. The provided elements drive the definition of the objects required for the implementation of the application with little effort with respect to other languages. In this manner it is possible to develop a demonstrator for certain methodologies in less time with respect to other choices. In particular the provided instruments allow to create and clearly define the back-end structures without a direct coding by the user since this process is managed separately by proper functions. In this way the developer does not necessarily have to know all the features of a web-based architecture for a certain application but can work mainly on other aspects as the correct definition of the relationships between the models that are then integrated in the database schema. In the same manner more time is left for the creation of visualization and interaction utilities with respect to the client-side interface.
Ruby on Rails offers also some interesting capabilities for the management of database information during the development phase. This feature allows to test different implementation alternatives and choices with minor code changes. In this way it is possible to quickly evaluate different solutions with the final aim to identify the configuration that show better behavior. The implemented framework has been developed mainly to assess the feasibility of such approach but some performances improvements can be obtained with the implementation of other languages on different platforms.
In particular the implementation of Ruby on Rails web application has also been developed following the main features of AJAX philosophy. RoR environment allows in fact for a well-organized integration of JavaScript, CoffeeScript and jQuery functionalities, providing useful instruments for the server side communication. These elements are mainly introduced to improve the performances and navigability characteristics of the developed framework. In this way the server is interrogated only for the strictly required times, avoiding the reloading of page for the same objects that have already been loaded. Some of the requests and responses between server and client has been based on JASON exchange format, exploiting a lighter communication for the available resources. The same result can be accomplished through the use of other exchange formats as for example XML files. The previously introduced languages in the context of AJAX paradigm are briefly described in the following lines.
jQuery is mainly used to define well defined calling for the server, avoiding the need to refresh all the page. JavaScript is instead used to run some executable functionality on client side, avoiding the need to call the server for something that can also be accomplished without its direct intervention. In this way it is possible to create a more dynamic page navigation, reducing the time delays related to server interaction. CoffeeScript has the same function of JavaScript but allows a better code organization through a more synthetic definition. JavaScript has also been used to implement those functionalities related to the information export, like for example the generation of images from the diagrams defined during the modeling activities (through the methods and event management provided by the previously introduced languages).

7.4.2 Conceptual overview

The definition of the conceptual model used within this work follows the current proposed meta-model for the management and formalization of the engineering data information. The conceptual classes introduced have been considered concurrently with an analysis of the possible integration with the design development process. In this phase it is important not only to consider the potential benefits of a certain model-based approach for the interfacing of the design and analysis methodologies but also to evaluate an effective integration with the actual modeling techniques. The main scope is represented not only by the evaluation of a model-based methodology for the management of the information related to the design and development process but also by a clear understanding of the potential improvements that can be gained by all the people that are involved in the system design. The objective is to evaluate the feasibility

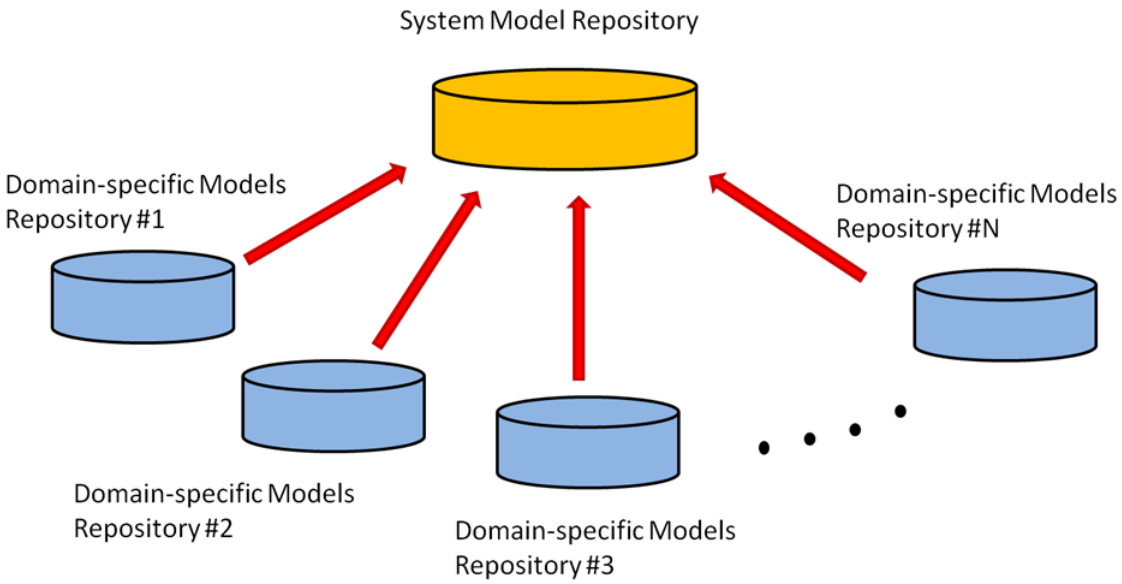

Figure 7.9: Example storing strategy for the management of project data.

of such an approach through the implementation and improvement of a currently under development collaborative tool. This one may not be barely imposed to all the different engineering personalities involved within a project but the ideal solution is represented by the development of an instrument that can be accessed and used without particular competencies, avoiding the need to train all the people involved.
On the basis of the previous considerations the integration and implementation of such a methodology has evolved concurrently with the study and analysis of the current pattern for the storing and management of all the information related to a certain project. This phase is fundamental for the right understanding about the actual technologies used to manage information on system model and how to integrate the proposed methodology with this ones. The generation of a theoretically perfect tool that is not well suited with the integration of the considered technologies is not a good solution. A study on the actual used technologies and the conceptual definition of the proposed methodology need to be considered concurrently.
A very simple representation of the storing pattern for the proposed approach is reported in figure 7.9. This figure shows a preliminary and indicative architecture, introducing the main environments that are described with more details in the following lines.

The previous figure represents synthetically how the model repositories are considered for the definition of the proposed methodology. The system model is stored in a central common repository, following the guidelines of the model-based paradigm. The presented architecture is only a conceptual representation since the system model repository may be physically represented for example by multiple repositories for redundancy. In the same way the repositories that store the models referring to the domain specific environments (for instance thermal domain, structural domain, etc…) may be physically identifiable in multiple repository spread over different servers or workstations. The actual architecture is also strictly related to the people that have to manage the information and work on the available data. System engineers have mainly to face with the system model while domain experts (domain specialists) have to work on domain-specific environments with their related tools. System engineers generally follow a what can be defined as a top-down process where the system requirements are managed to build the design of the product while domain experts work often on the definition of simulation and analysis models. The objective of the analysis and simulation models is substantially the verification of requirements. In this case the definition process follows what can be defined as a sort of bottom-up process where the actual models (generally the most reliable models in the current development phase) are compared to the specifications coming from the system model to assess the actual development level. In this way it is possible to assess the maturity of the system with respect to the target configuration and also to discuss about the possible changes that can be introduced to increase system performances. As the development process proceeds the ideal situation is represented by the convergence between the target design and current implemented one (this

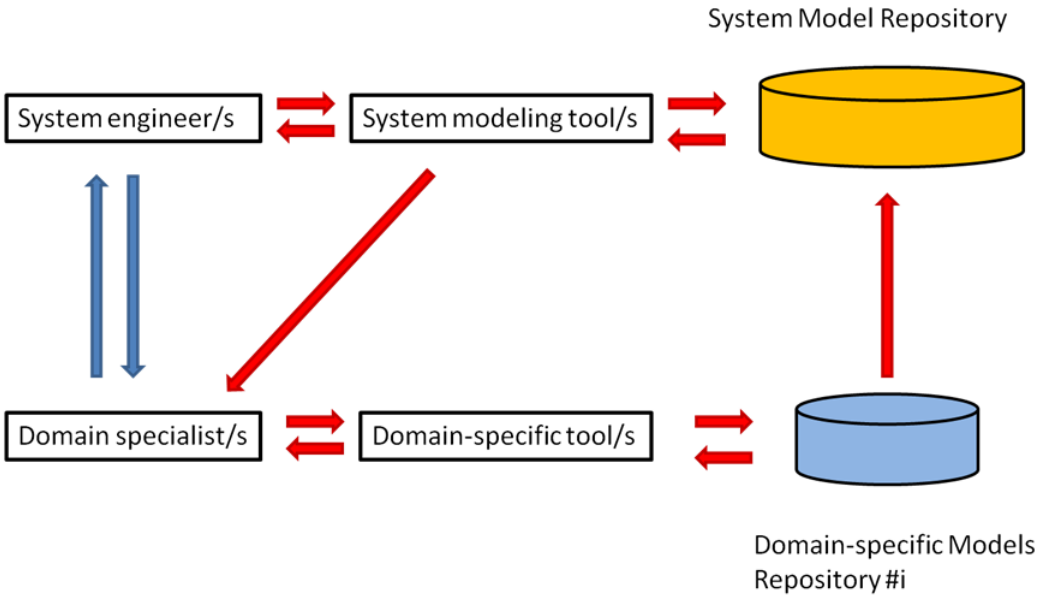

Figure 7.10: Example representation of the possible solution for the management of data among system engineers and domain specialists.

latter one is represented by the increasing fidelity models and also the possible produced elements as the system goes towards the acceptance phases). The distinction between the system engineers and domain specialists (considering the processes and the people involved) is not always so clear and the considered classes are often merged. The previously considerations must be taken as conceptual guidelines to understand the meaning of the proposed methodology.
The intercommunications between the various domain-specific models with the central system model are realized above all through the definition of a standard communication protocol where the data exchanges can be provided with ad hoc adapters, depending on the interface to be implemented. A simple and conceptual visualization of the relationships between the different data repositories is reported in figure 7.10 where also an indicative representation of the people involved are represented. Only one domain specific environment is visualized for clearness reason but the same identical connections can be identified in all the possible domains involved.

The previous figure represents briefly a conceptual visualization for the intercommunication links showing also the main features used for the exchange of information between all the involved roles. The system modeling tool may be represented generally by different solutions. SysML tool is currently under investigation to understand its capability to model all the possible characteristics of a certain project from different viewpoints depending on the involved design domains. Other solution may be represented by the development of other tools in other languages with similar capabilities as also web-based services that can be used to define the proposed methodology. In the case of the domain-specific tool the instrument can be represented by the standard de facto tools used respectively (for example Nastran Patran for the structural environment).
In the proposed process pattern for data exchange different communication rules may be adopted on the basis of the specific needs for a certain design process. For example the communication between the domain specific repository and system model one is only in one direction and physically implemented through the use of adapter element. In this case the information can be extracted from the domain-specific environment and loaded within the model system repository without over-writing the already contained data. The data needed by the domain engineers are gathered from the system modeling tool and the actual characteristics of this link can be defined on the basis of different needs. For example this connection can foresee the possibility for the domain engineers to access the information of only a filtered area depending on the related credentials and also they cannot edit some resources except for instance some form of annotation (useful for the management of the historical information changes and considerations).
The proposed modeling methodology theoretically allows the generation and definition of design options

(as the design parameters identification) only for the system engineers only (considering also the domain system specialists for each domains). This approach does not deny the possibility that the design options and variables can be proposed and identified also by the domain specialists. The investigated approach allow however the access and modification of these feature only to the system engineers from a theoretical point of view. This consideration is however a conceptual definition for a formal characterization for the modeling approach but similar and slightly equal process for the modeling priorities can be considered and assessed.
The main idea is however represented by the possibility to take account for such modifications in order to maintain a clear view of the overall system project. One of the main advantages of a model-based methodology and from this viewpoint also the proposed one is represented by the capability to keep information about the traceability of the modeled elements and components.
The data exchanged between domain-specific repository and the system model one can be represented by file such as models or result files coming from simulations. The main idea is also represented by the possibility to load some preliminary models to directly allow early analyses on the basis of simple models that can run during collaborative sessions. This capability can be obtained binding this domain-specific models within the system model repository (for example within a file system directories on the same server-machine or however on the same conceptual system model repository however). In this way may be possible to distinguish between the current data available (stored for example in a database file/files) and the model (also if simple) related to the system model. These objects can be stored as resources for a certain element loaded and defined within the database file. This technical approach is related to the files dimension since a particularly big file can be difficult to store in the system model repository with potentially other big files coming from other disciplines. A right balance is represented by the possibility to store some simple and light models within the same system model repository to provide a useful support for early and collaborative sessions. This functionality can be useful during the preliminary design phases where multiple design configurations are analyzed to understand the feasibility of a particular solution. The models generated for a particular analysis can also be traced as resources through the definition of the related file system path to the related domain-specific repository. The same components as modeled within the system model repository can be generally related to different discipline-specific environments, each of which can be characterized by different models (always for the same component), each of which is related to a particular analysis to be realized or fidelity, depending on the development level and progresses during the design development.
The idea is to conceptually relate the representative model of some particular object to multiple analysis models, coming for example from different disciplines. In the case of the same discipline different models can be built due to the different analyses that such a models have to be defined for. This concept is mainly related and represented by the class of Element Occurrence that has been introduced with the concept coming from ECSS standard. The idea is to relate the same element Definition to multiple analyses models that refer to the same conceptual class.
One interesting feature for the previous considerations is represented by the possibility to group the generated elements within a library repository for reusability purposes. In this way the elements that have been created for other purpose can be reused within another context and project with reduced modifications on the already available elements.
The previous concepts may be extended on the basis of the actual implementation, taking into account the possible solutions with respect to the realization of a collaborative environment. The concepts considered until now for the implementation of a collaborative environment can in fact be actually integrated through different quite similar solutions. In particular various alternatives have been conceptually considered but the target of the current work is mainly addressed towards the investigation of the methodology and the data structure. The proposed data structure is evaluated to understand if such approach allows the management of the design and analysis scenarios that have been considered as reference cases. For this reason only one alternative of the possible architectures has been actually implemented. Other solutions can however be approached on the basis of specific needs and available resources.
The considered infrastructures are conceptually reported in the following figures and they are the result of our considerations but are not necessarily the only possible options. Each solution has its own features,

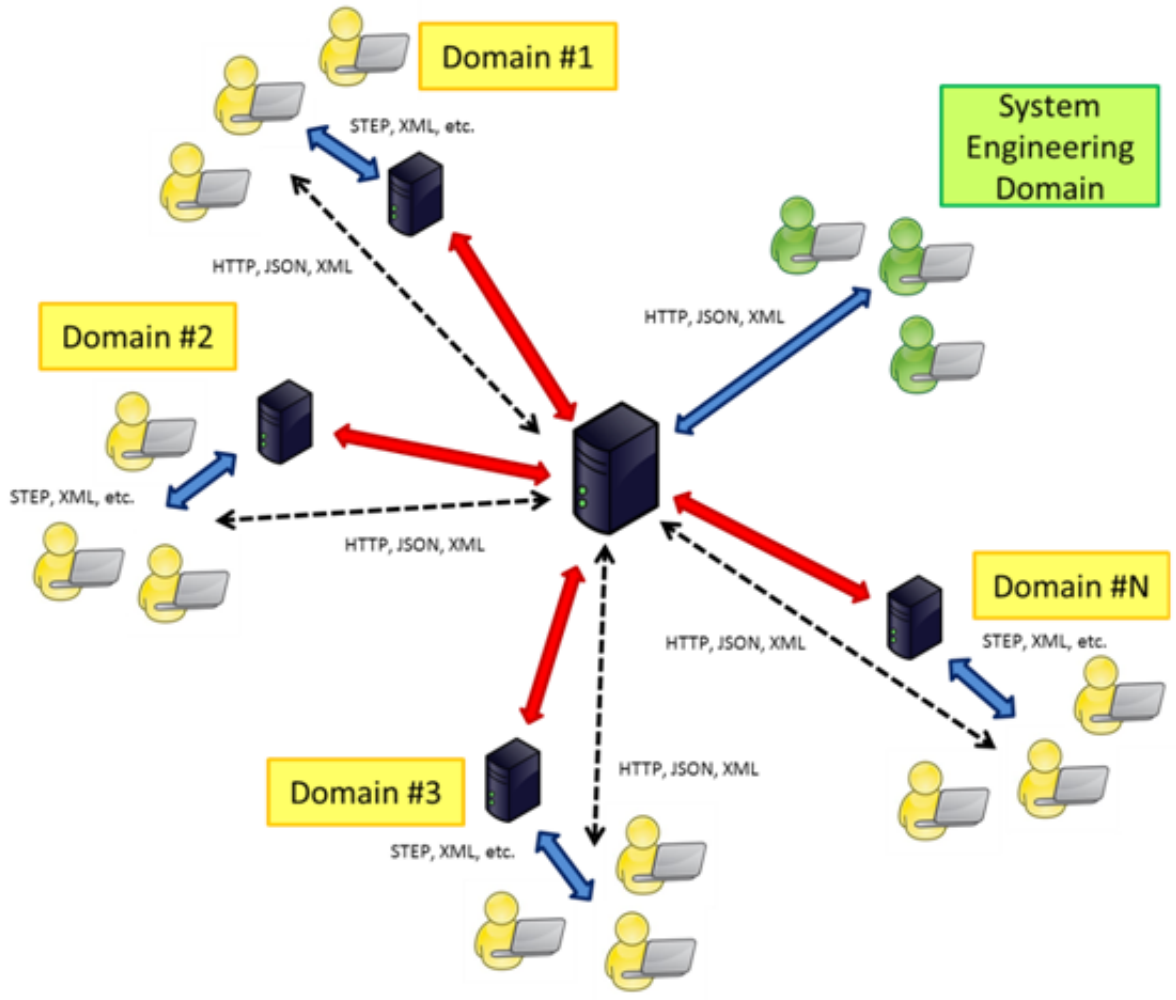

Figure 7.11: Conceptual overview of the infrastructure for the collaborative framework.

benefits and drawbacks with respect to each other.

The configuration above (7.11) considers the presence of a web-based platform for the management of system model information, directly accessed by the System Engineering team. Such application is however conceived to provide information to all the involved domains, enhancing the collaboration and design coordination (workflow scheduling, data consistency and sharing, etc.). Other web-based platform can be considered to support data and models coming directly from analysis tools for each domain (structural, thermal, etc.).

These web-services are used to manage analysis models generated from disciplines-specific tools. The information created and collected in this way from different environments can then be processed on server side to be shared with a common system engineering platform. Each user belonging to specific domain uses the tools already validated and available for the specific analysis environment. They store the information and files on their local machines (where they also work for the current baseline data) while files and information to be shared can be uploaded or edited through dedicated web tools. Each user can access information coming from other domains or users on the basis of his/her role within a specific project. All such data are mainly provided through HPPT protocol, suing JSON or XML format for example. The web application that is directly used to manage system engineering data is conceived to potentially communicate with the other domain specific platform, reducing the time spent on data exchange in a collaborative environment. In this way it is also enhanced the capability to monitor the consistency of exchanged information, with the possibility to partially automate the model transformation process through proper developed routines. The server machines reported in the graphical representations are the physical location of the web-based applications and the related database files. They can however be associated to file-system resources to store all the information and more properly those files that cannot be directly up-

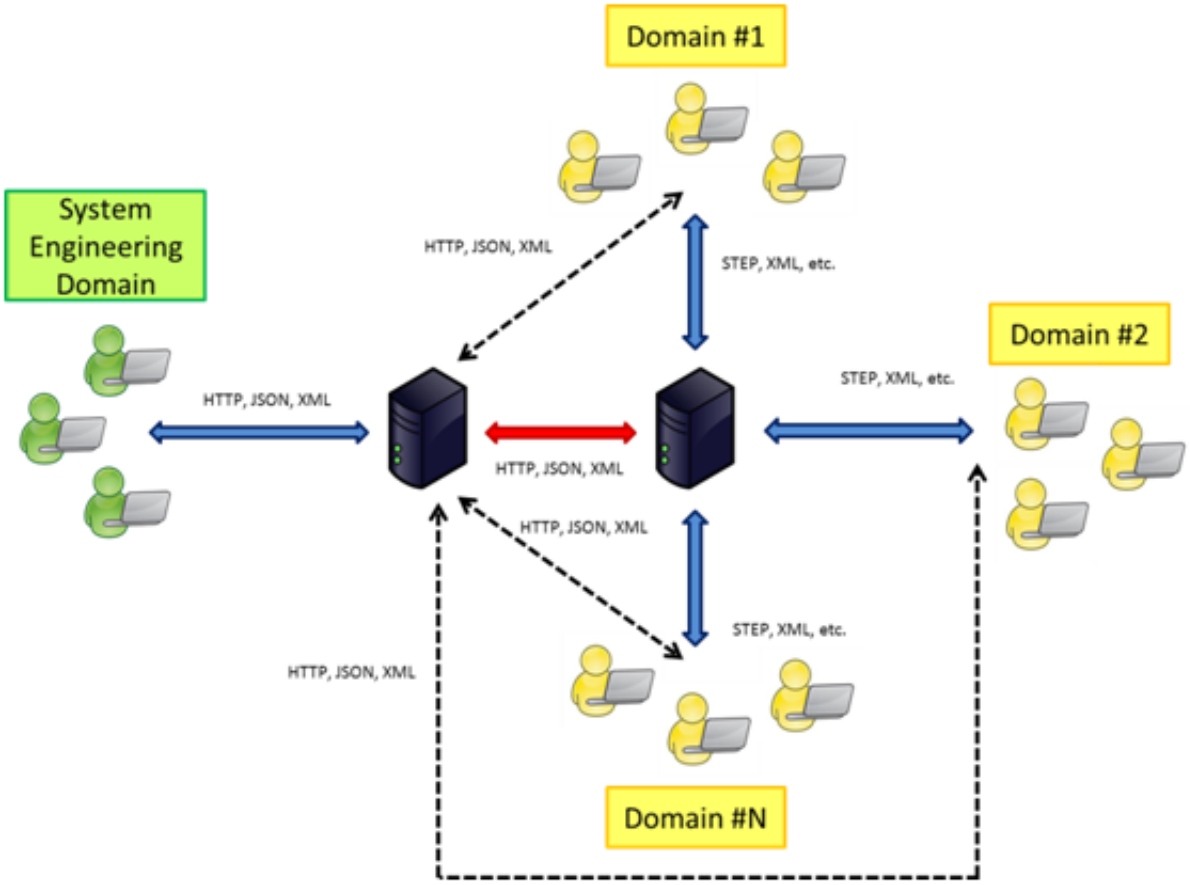

Figure 7.12: Conceptual overview of the infrastructure for the collaborative framework.

loaded within database files (as for example images, text files, 3D models, etc.) but that cover a key-role for overall web-services (a 3D model can be used in fact to provide some visualization capability through browser interfaces, for example loading .wrl models). In this case such resources not necessarily must be placed on the same physical machines that are hosting the web-services but can also be stored on a file-system on an accessible network (in the case for example the required files are place physically on another machine). These last observations are valid also for the next architecture types.
The configuration discussed first requires the definition and implementation of all the involved web-based platforms for the various domains. The management capabilities required to process the files coming from different sources (different analysis and modeling environments) are quite similar across each discipline. The formats of the elaborated files will be different (depending on the related modeling and analysis tools) but the processing activities performed on them are the same. Considering such aspect the web-platforms that handle the files coming from the various disciplines can be ideally unified in one framework, maintaining however a dedicated application for the system engineering domain. From these observations another conceptual architecture can also be considered. The related representation is reported in figure 7.12.

This configuration shows an architecture type quite similar to the previous one. The concepts introduced above are the same for this kind of infrastructure. The main difference is due to the fact that the web-platform for the management of domain specific resources is the same for each discipline excluding the System Engineering one. Dataflow among the various actors is however based on HTTP protocol and XML or JSON formats. Other file extension as the STEP one is however used within a specific domain to exchange data with the web-based platform for example. Processing utilities can be properly set to manage the files coming from different domains, exploiting the STEP file format to formalize the data contained within a particular model. Following this standard the elaboration of the information available within a certain file can be better captured, processed and stored in a consistent and formal way.
As in the previous configuration one of the main ideas of such web-services integration is also represented by the fact that some information can potentially be requested by the web-platform of System Engineering domain, calling directly the web application that manage the files of the various domains.

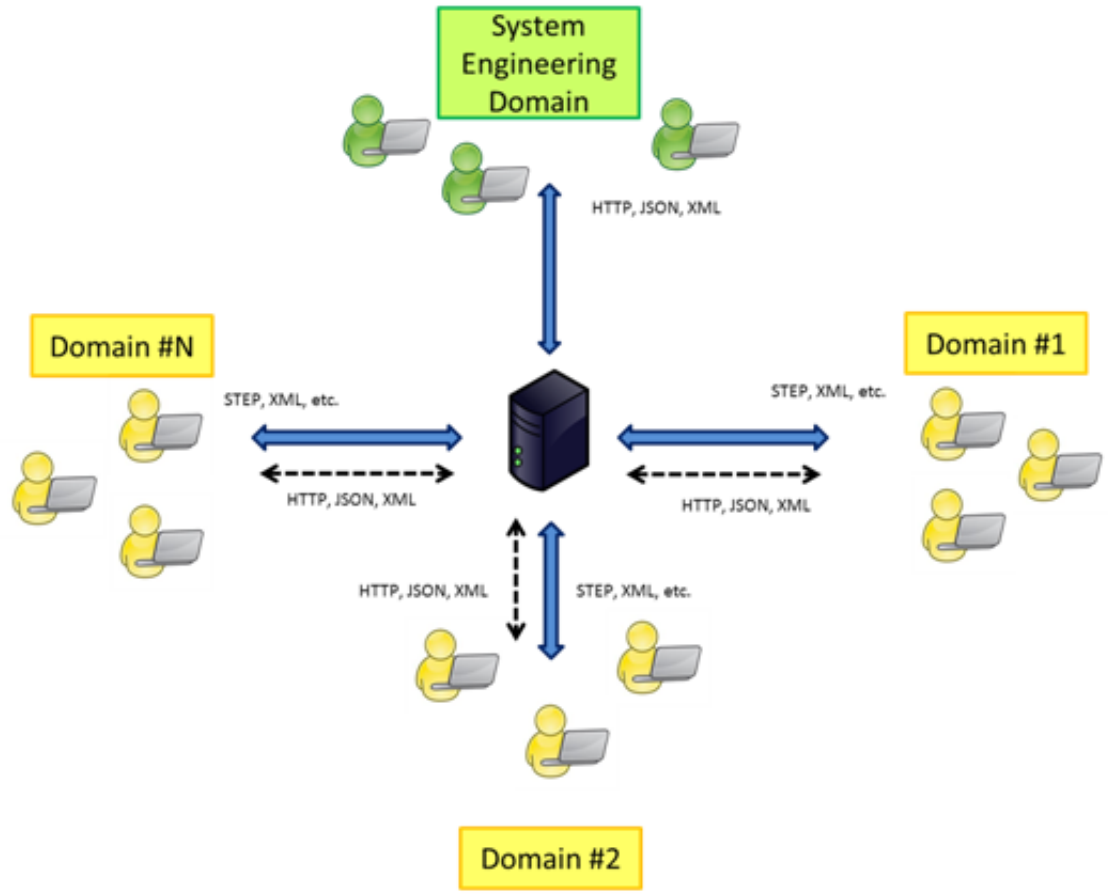

Figure 7.13: Conceptual overview of the infrastructure for the collaborative framework.

The distinction between the web-service related to the System Engineering domain and the other that manage the remaining disciplines is basically done to better distribute the available resources. This solution requires however the implementation of two distinct web-based tool for the two different purposes. From this viewpoint a simple implementation can be achieved through the use of a unique web-based platform, for the activities directly related to the System Engineering domain as well as for the management of files coming from modeling and analysis domain-specific tools. In this way the capabilities will be provided by the same service while the resources that need to be stored on the file-system (images, text files, 3D models, etc.) must be placed where the application has access capabilities (for example the same company network). A conceptual outline for such architecture is represented in figure 7.13.

In this configuration the overall platform is implemented on the same machine, providing the functionalities of all the involved domains. The framework is basically conceived to support the activities directly related to System Engineering domain while the other domains are managed in a different manner. In particular the main part of modeling and analysis activities of these domains is managed within each discipline environment through dedicated tools. Only a filtered set of data are instead processed to be shared, compared and visualized as system model data. In this way data surplus is reduced, paving the way for a more effective management of the project information. The implementation of a unique web-based platform allows e better maintenance of the infrastructure. The integration with domain specific environments is pursued through the generation of dedicated adapters directly embedded within the application and that are used to process the uploaded data. The information loaded from a System Engineering perspective must be properly filtered and processed to avoid too much data shared across the project. At the same time the information collected from the various domains must be properly shown to the other actors.

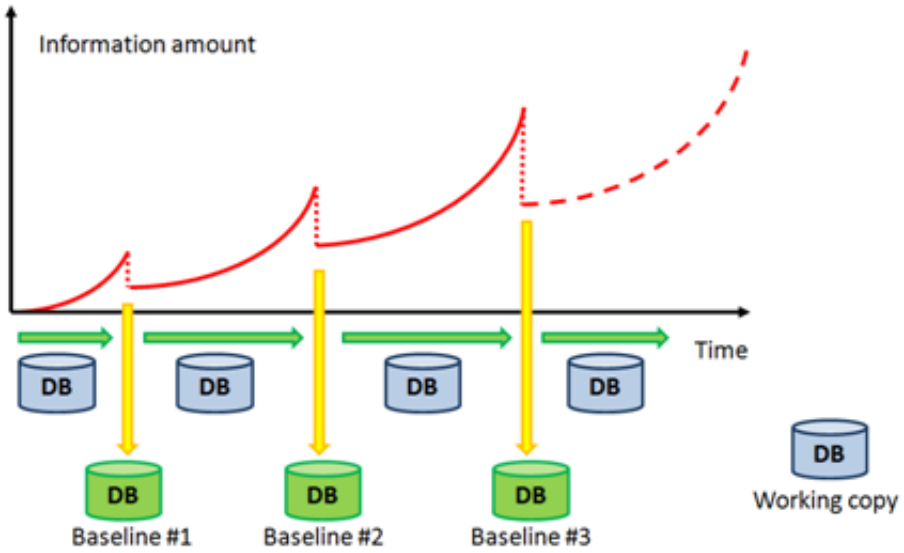

Figure 7.14: Example storing strategy for the proposed architecture.

7.4.3 Requirements management

The management of project requirements is another fundamental aspect for the correct design and analysis of a complex aerospace system. The implemented framework has also been conceived to be integrated also with external tools already available on the market. A widespread commercial solution in aerospace industry is represented by IBM Rational DOORS (IBM Rational Dynamic Object Oriented Requirements System). Such tool is a client server application, with a Windows-only client and servers for Linux, Windows, Solaris and is based on dedicated programming language called DOORS eXtension Language (DXL). The interfacing with external environments can be achieved through ReqIF (RIF – Requirements Interchange Format) file which is basically an XML file that can be used to exchange information and associated metadata with other requirements repositories. The same interface format can also be potentially used to integrate a dedicated adapter for the data exchange with the proposed infrastructure. Future developments will consider such connection, evaluating the capability to easily share the requirements with the partners working on the same project for example.

7.4.4 Baseline and database integration

The proposed modeling framework has also been conceived to be well integrated with a demanding need of data storing as the system details increase during the development process. This considerations has lead to an analysis of the current methodologies for the management of system information and the related procedures within a consolidated design procedure. In particular the meta-model data structure has been conceived considering also the concepts related to baseline formal definitions and their roles during the development process. Baseline data are properly stored consistently with the database resources, providing at the same time the starting point for the development of the current working copy of system design. An example storing strategy for the proposed architecture is conceptually reported in figure 7.14.

As the proposed framework has been conceived to trace all the changes operated on the system (author, time and value of changes) the data contained can rapidly increase. System information are saved (changes tracing included) when a baseline configuration has been established but the project development in the following working phase starts from the data contained without all the information gathered for the changes. In this way the database for the new working copy is not so heavy but the historical information are however stored (in the case for example some review are needed on old modifications). In this manner the starting point for the project design can access all the required information, avoiding an overflow of data. This working copy then store the changes related to its phase until the following baseline.

7.4.5 Diagram generation and management

The integration of visual capabilities within the framework has been developed starting from the development of interface with external plug-ins. For example the VRML files are managed through the use of Cortona3D (www.cortona3d.com), a plugin which provides a complete 3D navigational tool-set allowing you to rotate 3d models and to walk within 3D polygon-based worlds. In this way it is possible to create a dynamic navigation of the object loaded within the environment, supporting the activities related to the analysis of system and subsystem configuration. CAD files can then be extracted from the available commercial tool, exporting for example the VRML data and then properly loaded in the same modeling environment.
Other extensions are considered for the integration with a web based application like for example X3D. In the future this format will be better integrated within HTML5 protocol with the main aim to improve object embedding with web applications. Currently some efforts are addressed towards a wide formalization for the management of graphs, 3d objects and generally visual elements with canvas elements, rendering directly the scenes and all the required data on the fly. The canvas element would soon become the placeholder for the Internet's next implementation into 3D, represented in particular by WebGL. This one is a low level programming language, providing a very technical code and without a layer of helpful abstraction to support the user in reading and learning the syntax. For this reason a good number of developers have already begun producing tools and JavaScript libraries to support scene generation, increasing the accessibility of WebGL.

7.4.6 Tools, languages and development platforms

In the next lines a brief introduction of the tools, languages and development platforms that have been considered is presented. The main features, benefits and drawbacks of various alternatives are described through a little overview. The main scope of such introduction is to assess and highlight the main reasons for the selection of the available options. For example some languages offer advantages not available in other ones but at the same time these last show benefits that cannot be find in the other. This situation underlines again how the main aim of the current work is addressed towards the demonstration of the proposed infrastructure. The final implementation depends in fact on specific implementation backgrounds and resources availability.

7.4.7 Description on the benefits and advantages of open-source tools.

The number of projects and research teams involved in the development of open-source tools is constantly increasing. Nowadays there are various initiatives that provide services and instruments to the world community. They often arise from university research teams or within a restricted groups of people animated by common topics. The projects are usually self-sustained through the whole users community after the first version publicly released. Hence the software improvements, de-bugging activities and documentation are ensured by the user contributions. The main advantages is represented generally by the absence of a commercial licenses required for the execution of the tools and all the possible problems related for example to the network management or updating of these ones.
Open-source tools can be instead quickly downloaded and installed without particular problems. There are also specific functionalities that allow to properly manage the updating of the current version used, highlighting the simple upgrade as an interesting feature.
Another attractive characteristic of these initiatives is represented by the accessibility to the code implementation. In this way it may be possible to properly customize the software for the specific needs and company knowledge, without the constraints of commercial products.
Open-source initiatives and projects are often characterized by not well developed interfaces and the integration with user provided codes requires consequently the implementation of dedicated solutions. User friendly interfaces are often not available while communities developed modules and libraries are not rigorously documented in certain cases. For this reason the learning curves related to open-source languages

and platforms are often lower than commercial solutions as the tools or languages are approached for the first time. The main advantages of open-source infrastructures become evident as the users spent increasing time with them, developing the capability to achieve an high level of customization. Another drawback of open-source projects can be represented by the fact that some initiatives dismiss their maintenance and developments, not more providing support to other projects based on the same infrastructure. Such maintenance and developments in other cases rely on small group of developers that generally work on other different subjects and all the corrections or debugging activities are done with no scheduled procedures.
Some research initiatives are currently addressing their efforts towards the development of Open-Source framework for multidisciplinary analysis and optimization. This aspect further underlines how the available tools and technologies started to provide interesting capabilities with respect to system analysis and design. From this viewpoint interesting results are available in [110].

Ruby on Rails

Ruby on Rails is an open-source web development framework that has been conceived to reduce the workload that the programmer must do to implement the desired applications. In particular it provides some useful capabilities that can help the programmer to set up the required infrastructures. Some code sections are in fact automatically build from simple command line instructions that generate code portions that include all the basic elements required the application backbone. The related development environment favors convention over configuration paradigm. This definition refers to the fact that following the convention supported by Ruby on Rails infrastructure it is possible to automatically generate files, code portions and settings that otherwise must be manually defined. Such elements are basically built on the basis of convention patterns that is also used to automatically define the relationships that properly connect all the created elements. In this manner it is possible to quickly implement and evaluate a web application to assess its feasibility without spending too much efforts on applications basic scheme. This capability is achieved through the convention paradigm that ensures the reduction of consistency problems (among code portions) and time required to update the applications after some structural changes. This approach allows to increase the productivity, enhancing also the debugging activity during the development process since the applications changes can be easily managed. Another interesting feature of RoR is represented by the fact that such framework has been conceived to follows the main guidelines of Agile Web Development lifecycle methodology. This approach allows to improve the development, test and production activities required for the release of a web application as previously introduced (more details can be found in [84]) and [85]
All these features lead the choice of RoR development framework concurrently with the fact that a large number of libraries, forums and well-documented functions are available from internet. In this way it was possible to mainly focus the efforts on the methodology and conceptual infrastructure while the web-service has been implemented only to assess the developed approach and meta-model. The same result achieved with RoR framework can in fact be pursued with other solutions but it is not the target of the current work.

Python language

Python represents one of the most widespread scripting language used for communication and execution purposes above all for its cross platform capability to run independently about the operating system used. Python language comes embedded with most of the current Linux distributions and installers are however available for other operating system such as Windows or Mac. The capability to run Python scripts on operating system different from the native one (the one where the script has been implemented), or however with minimal changes of the code, has made easy the spreading of such language within engineering applications. This feature is directly related to the nature of interpreted language of Python. In this case the code does not need to be compiled before the execution and the code lines are elaborated to machine level one after the other. In this way the code is interpreted each time it is executed on the basis of the current machine where it has been launched. On the other side the compiled languages are optimized

for the hardware machine where they are executed to create the executable. In this way the executable generated is strictly bounded to this machine while the Python code is not directly related to the hardware of the machine where it is launched. The advantages of compiled codes is that they are optimized in the generation of the machine level code and in this way the performance that are reached are higher than those obtained with interpreted languages. In particular compiled codes can reach different times the execution speed of that shown by interpreted ones. Code that required the execution of long time simulation or however that have to be called different times during the simulation campaign are typically written with compiled language (as for example C++ language). Different research initiatives can be found in literature that exploit the advantages of Python for the management of simulation codes (an interesting example is represented in [108] where a Python based infrastructure is used to manage CFD computations)
Python language shows a valid alternative for the implementation of simulation interface functionalities in the context of DAKOTA framework. This scripting language, as will be described with more details in the following sections, is one of the supported code for the integration of DAKOTA simulation toolkit with external solver/applications. This choice is related to the wide spreading of this type of language for automating simulations and manage data exchanges between different platforms. Other scripting languages can be considered for this kind of integration but their use is not well documented and tested. For example Perl or C-shell scripts can also be used with the same purpose of Python language.
In particular Python has been chosen in this work has the scripting language for the management of the models and simulations in the back-end perspective of the implemented framework. It has been used to define the filters and drivers that allow to process the data available from DAKOTA environment, providing the right formats for the specific external solvers/codes. In this case the scripts belong to the family of input filters for the framework data flow but more details will be provided in the sections related to the effective code implementation of the framework itself. In the same manner once the external solvers/codes ended their executions the results obtained need to be processed to provide the correct formats to pass again to DAKOTA framework. This function is performed by Python scripts in a similar manner to the generation of input filters but in now they are called output filters to distinguish from them from the previous ones. Their roles are however quite similar but the main scope of the elaborating process are different from one context to the other.
The Python scripts defined in this way are then properly integrated in the main driver that manage the execution of the single iteration within the context of analysis cycle.

Django web development framework

One of the web development frameworks that are available for the definition and testing of web-based applications is represented by the Django initiative. Django framework is based on the Python language and it provides a series of functions and API that facilitate and automate the development of the required application characteristics. In particular the instruments available support the creation of the web-application architecture. They are used to set and properly define the creation of the link between for example the URL page requirements and page controller for the rendering of the requested information.
Part of the current work has considered the use of such modeling environment for the development of the user interfaces needed. The integration of the DAKOTA functionalities through the implementation of web-service with Django seemed to be the proper choice, following directly from the fact that Python is also the language used for the definition of filter and driver for the simulation codes. In this way the time spent for the definition of the required interface seemed to offer the advantages to develop with the same language the required features. This approach has shown some implementation difficulties related to the definition of the application, regarding above all the management of the database correction. Since Django environment is an open-source environment some features are not well defined and are currently themselves under development and also not so well documented functions. In particular the definition of model classes for the definition of the elements that have to be introduced within the system model is currently strictly related to the database population. This characteristics mainly affect a try and error process for the correct insertion of records within the database file. In particular each time the model classes have to be changed the database must be reset and repopulated again, ensuring that the table's columns

are properly corrected (during the early development phases the model itself may regularly be changed and the operation of reset the database may be particularly unproductive). This operation may be done several time during the development phase of the application, driving to a not well effective process. Different plug-ins or initiatives are trying to correct this aspect but they are not particularly stables, affecting the overall implementation process. Other open-source development frameworks are not characterized by this deficiency and they are more suited for the development of applications also during a conceptual definition phase. Ruby on Rails is for example one of the web-development framework that allow to face this problems separating the generation of database file (and updating the related schema file) from the model classes codes (using the migration files for the correct upgrading of the database characteristics). In the following implementation phase for the definition of the web-application tool Ruby on Rails has been evaluated as the most suited instrument for the definition of the demonstrator for the current study.

Modelica language

Modelica language is the other interesting open-source resources that has been considered for the implementation of the modeling and simulation features of the developed demonstrator of this work. In particular Modelica has been chosen as a well suited language for the modeling of some simulation blocks. One of its advantage is mainly related to the possibility to build models where input-output relationships are resolved just before the simulation execution (during the checking process). This capability allows to model a particular element without constraints on the directions of input-output variables since these are computed by the Modelica solver itself. This feature is not available in other simulation environment as code write in Matlab or within Simulink environment where the relationships between the input and output variables must be known before the modeling activities. In this case the variables that are defined as input and the ones that are instead the output must be known before the simulation itself. Modelica models are instead characterized by a more physical-based definition of simulated objects. In this way the same model can be simulated within a different scenario without not so extended changes. On the other side models characterized by a well-defined relationships between the input and output flow must be deeply modified to satisfy the new defined scenario conditions. The reusability of such models are not so advantageous as that related to the implementation with Modelica language. Model modularity is one of the other interesting characteristics provided by Modelica with respect to other modeling languages where the modularity is not obtained with the same level of depth.
From the previously considerations it is possible to foreseen how networking applications show interesting characteristics in the context of a collaborative architecture. In particular such approach can considerably improve the collaboration between different people involved in the same project with the futuristic vision to extend the advantages of a model based approach not only in the first phases of the development but also in the following more detailed ones. This objective can be reached with the introduction of the proper instruments in the design process, considering also the need to partially modify the current development procedures to account for such changes. The design process must be partially changed for the implementation of model based methodologies that can allow to improve the overall system performances thanks above all to a better organized architecture for the system definition.
Networking is currently one of the most challenging feature that has start to play a key-role also in the simulation environment as for example in the implementation of ad hoc interfaces that manage the requests routing towards a common central model. Different simulation software houses in the last few years have invested a lot of resources for the integration of web-based services. Simulation models are for example stored in a central repository that is often directly connected with multi-processors/multi-cores computing resources (generally strictly dedicated to the solving characteristics of the models features) while the users that need the results from such models exploit the information provided by a web services. In this manner it is possible to decouple the computing resources (particularly cost demanding and often used only for a little period during the development process) from the users' needs (that for example interrogate the same model for certain information only for a reduced time with respect to the development process). This scenario is for example currently identifiable with some structural solver suites that provide the computing resources for a certain model that is interrogated through web applications but is not maintained

by the same user (in the same manner the costs related to the management of the computing resources are allocated to the computing services that works for different industries and organization at the same time).

MathML

MathML has been conceived as an application of XML with the main purpose to describe mathematical notation, formalizing both the structure and content of a certain expression. This language is particularly suited for the exchange of mathematical data across web services. Its capabilities are particularly interesting for the main purpose of the proposed approach but its direct application has not been considered in this work. The integration of such language within the developed framework can potentially be considered in the future.

COM Interfaces

COM interfaces represent one of the features that can be used to manage the information exchange across different applications on the same or different platforms. This solution can be exploited to allow the communication between different software and environments, ensuring also the possibility to proper elaborate the data gathered from various sources.
The model creation from math is now one of the most interesting research topics. In particular the information gathered within the system data model may be used to properly define its virtual representation. The development of this relation can be realized under different approach, depending on the required information. Modelica represents a well suited language for the characterization of the system behavior. For example the equations related to a particular element of the system could be used to set the physical laws that are successively used to manage the virtual simulation. The main issue concerns about the translation of the involved equations into useful codes that may be processed in the right way.

7.4.8 Design manager framework

The implementation of design functionalities has been based on the definition and concepts previously introduced and contained within the data model infrastructure. In this case RoR has been used to integrate the management of options and alternatives directly within the modeling application but similar tool can also be conceived as external framework. Web based architecture has been defined with the final aim to help and directly support the set-up of multidisciplinary design environment, paving the way for the integration of simulation capabilities within the same modeling environment. In this way it is directly possible to access the information gathered within the system model and to use such data for optimization cycle or trade-off study. The same information can also be passed through the proper web call on independent application but such integration would require more controls and checks for the consistency and synchronization of the available data and resources.
The integration of Python language within RoR application can be managed in different manners on the basis of the chosen approach. For example Python scripts can be called using Jython implementation. This one represents a Python implementation totally written in Java and the related classes can be managed consistently. In this way the standard modules available in Python distribution can be accessed in the same way as normal Java functions within the web framework.
Another way it is represented by the possibility to define a Python script with the required commands and then call that script from Ruby on Rails framework standard call for external function. This seems to be not the most effective since it requires different calls only to access the right interpreter. It must be more effective if the requests are directly managed and implemented in the same environment with only one interpreter language. For this reason the choice Ruby represents the most effective choice since the web application environment is implemented with the same environment.
Another family of integration solutions are represented by the possibility to exploit the web services capability that both this scripting language show.

7.4.9 Current implementation

A brief description of the current implementation of the conceptual architecture for the management of the same problems initially defined is introduced within this section. As other elements of the current data model all the definitions used within this study refer to the concepts available from the standard ECSS-E-TM-10-25. Some of the relationships defined within this technical memorandum has been slightly modified to take account for the integration with a design manager interface. The engineering model overview has been reported in figure 7.15.

The definitions of all the elements considered within this diagram refer the previously introduced taxonomy section. The concept of iteration has been slightly modified and in this work the evolution of the design project do not consider strictly the definition of an iteration object. The development project is handled in a more continuous way where milestone points identify periodically a common baseline that represents a formal recognized design level.
Iteration concept identifies in particular a representation of an iteration in the process of developing an Engineering Model. The iteration concept refers to the establishment of a complete and coherent step in the development of an Engineering Model. The identification of a complete and coherent design level depends too on the users viewpoints that can be different from case to case. This consideration has lead to the definition of a slightly different conceptual view and the related modeling approach.
In a concurrent design study the engineering model for the system-of-interest is developed in a number of iterations, where in each iteration the problem specification in the form of the Requirements Specification and a design solution in the form of the Options and Element Definitions are elaborated and refined. With an iteration the study team strives to set one more step in the direction of achieving a converged definition that fulfills the objectives of their study.
Has can be seen from the engineering model diagram the option object is connected with the Element Occurrence. In particular the Element Occurrence may belong to only one option while an option may contains no or multiple Element Occurrence. The option element belongs also to a certain iteration object and only one. At the same time instead the Iteration element can possess multiple options. The iteration objects belong also to the overall Engineering Model element. This approach has been slightly modified to take account for a different management of the options evaluations.
The Option class definition can be identified in the following specification (based on the standard meta-model). Option is a potential design solution for the system-of-interest being developed in an Engineering Model. An option in this context is a design alternative that can be compared with one or more other options, for example to perform a trade analysis.
The Concurrent Design Parameter that has been developed for the first time in the CDF meta-model for the first ECSS standard and that now has changed its definition is an interesting approach for the management of the project options. The concept of Concurrent Design Parameter shows an interesting solution for the management of the design options. The current definition for the Parameter object is a characteristic or property of an Element Definition. The concurrent design study centers around a multidisciplinary parametric design of the system of interest. Parameters (and their related values) assigned to Element Definitions, Element Usages and possibly Element Occurrence are the essential mechanism by which each Domain Expertise characterizes, quantifies, communicates and shares their part of the design with all other domains of expertise (expressed in the meta-model with the reference to the class Domain Of Expertise). The associated Parameter Type (through the parameter Type property) provides name, *shortName* and optionally alias, definition and hyperlink for this Parameter. This concept is the element closer to the desired element that can be considered in the proposed methodology for the management of the design activities. In the proposed conceptual framework a similar concept has been developed with slightly different characteristics and it is identified with *CDVariables* (Concurrent Design Variables) definition (this concept is strictly related to the concept of Options Group that will be defined in the following part). This object have theoretically the aim to represent a feature that belong to the Element Definition and can be shared among the various disciplines. The nature of this object shall have the capability to represent a property for a certain virtual Element Definition (virtual in the sense that not all its property are defined because the development phase is still ongoing and the Element Definition represents something not real). For

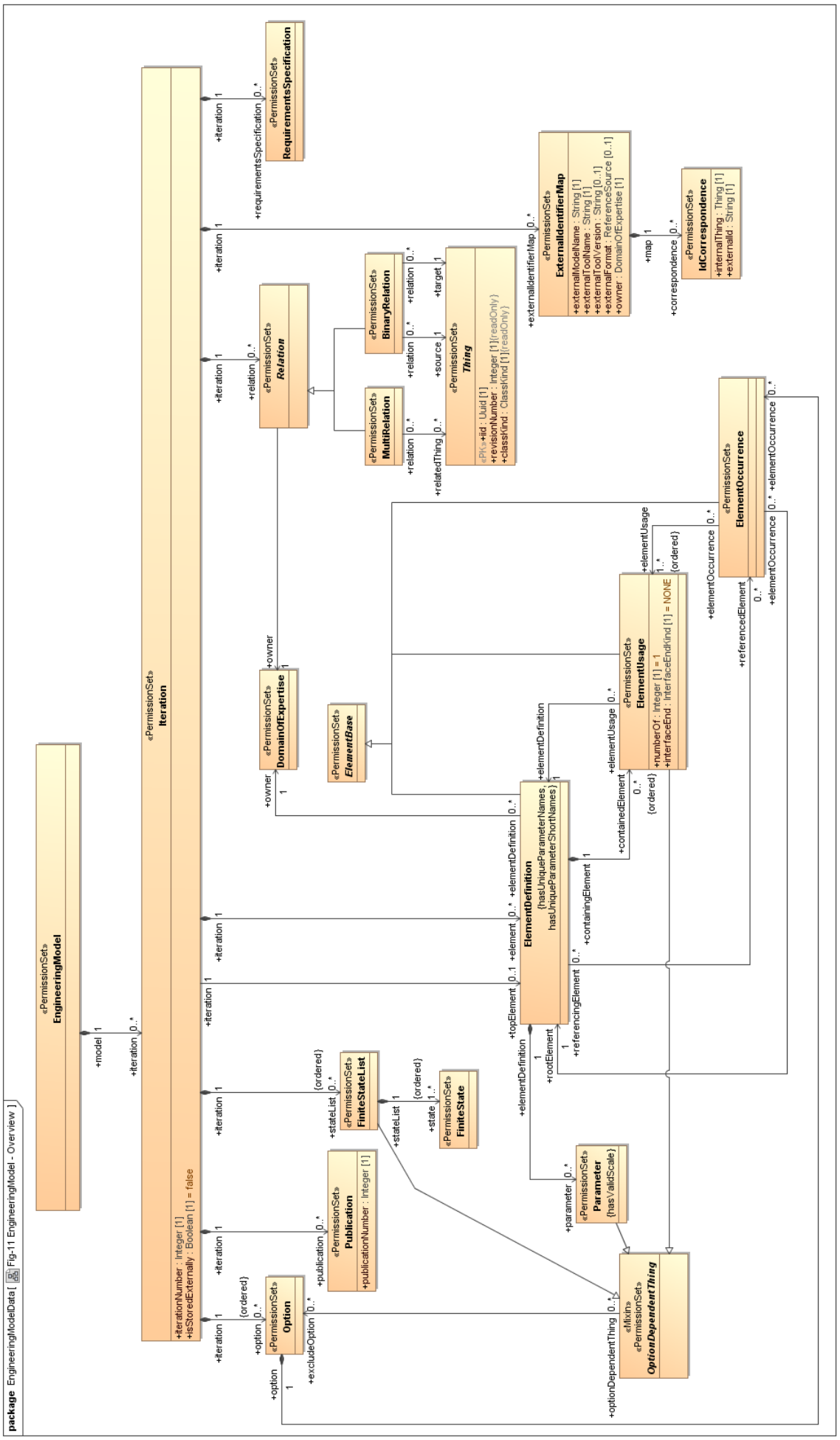

Figure 7.15: Engineering model overview, [61].

example this design variables can be related to a radius in the case of a motor-wheel or to a material property (considering the possibility to choose between aluminum or steel material solution for the production of the wheel element). It is important for this variable to be shared among the various domain-specific environments because its definition can affect various analysis. Still starting on the previous example it is possible to assert that the radius design variable can involve thermal, structural and mechanism disciplines. All the disciplines shall have the possibility to manage or however observe that this parameter has been defined as a design variable, since its modification can potentially involve their analyses. In other cases there is other types of design variables with no need to be shared with the other disciplines but that can be defined as design variables for a specific domain. In this case shall be possible to define this parameters as design variables but not to be shared with the other disciplines. In this way it is possible to trace all the possible design variables that characterize a certain project without sharing all the possible alternatives but only those that involve multiple disciplines at the same time. In the motor-wheel example the wheel element may be characterized by the definition of a small corner radius for the internal side of the wheel. This parameter may represents a design variable for the structural domain but not for the thermal one that have other fidelity requirements for the analyses models with respect to this parameter. In this case the Concurrent Design Variable is not shared with the other disciplines. This feature shall be implemented with the definition of a connection with the design parameter that can be associated from one to multiple Domain Of Expertise (or engineering domains). In this manner the single discipline can organize its design process on the basis of the filtered parameters, considering directly only those that are linked with it.
From a user point of view shall be possible to define the design parameter once the element has been created as Element Definition. In this way all the element Usage that depends on this definition all inherit coherently this Design Parameter but can also modify this values independently from other Element Usages of the same Element Definition. This object has to maintain independence from the other defined usages and potential relationships between two or more Element Usage with respect of the same design parameter have to be managed through the introduction of proper constraints between elements. For example the six wheels of a motor-wheel assembly (locomotion system) where the wheel definition leads to the presence of six wheels with six design variables radius (since for example in the wheel element the radius has been considered as design variable). Conceptually the proposed approach considered the independence of the six design parameters to allow in the future for the management of more complex situation where there is the need to singularly optimize the design for the various objects. In the case of the example instead there is however the need to impose the same radius for the feasibility of motion requirements. This constraint shall be modeled starting from the project requirements and must be checked with proper function or theoretically with automatic evaluation.
Another important feature that shall be implemented regards the definition for example of different range for the same design parameter of the same Element Definition. For example this case can be represented by the situation where the same Element Definition contains a certain design parameter (for example the length of a same type of battery) that in one case can be varied between certain range (for example the length can varies between 10 and 15) but in other part of the same system there is the need to insert an Element Usage with the same Element Definition but with the design variable with a different range (the battery length varying between 12 and 14). This situation shall be modeled defining a new Element Definition with equal properties but with different range for the Design Variables since it is better to manage all the possible changes from the Element Definition viewpoint. The Design Variable characterization (range, nominal value, in the future the mean and variance for aleatory variable, etc…) is managed in a more effective way if it is bounded to the Element Definition. In this way if we need to change the properties of the Design Variable we must generate another virtual Element Definition. This approach allows to better control the design process and all the design variables included within the project. If the Design Variable has to be managed (its characteristics as range, nominal value, etc) from the Element Usage perspective then the Element Definition can be only one but we have to access and modify all the range Element Usage for Element Usage if we want to modify for instance the ranges of the design variables. This approach need less work in the case all the Element Usage have the same range because we need to specify only one Element Definition and related Design Variable but if it is required to change the range of design variable considering different group of Element Usage then this require more work since all the Design Variables characteristics

(range, nominal value, etc.) might vary from group to group and all are independent between each other. If in this example the groups of Element Usage have been inherited from distinct Element Definition with different characteristics for the Design Variables then becomes more easy to change with less operations the design space properties.

7.4.10 Main features and realization aspects

The current design and modeling needs have been introduced in the previous lines while in the following section a more detailed description of the developed infrastructure will be provided.
A wide set of the most challenging engineering problems belong to the category of multidiscipline. Computational software using high-fidelity methods are well developed and validated for single domain. At the same time there is also a limited effort in the development and investigation of large-scale multidisciplinary analysis tools based on high-fidelity techniques. One of the most challenging activities is represented by the integration of hard formatted computational tools (coming from different disciplines for example) within a multidisciplinary environment. An effective identification of problems solutions is also closely affected by the use of high-performance computers and by common platform for resources sharing. Several high-performance computing initiatives have led to the implementation of very efficient but also specific disciplines codes. A very promising innovation is represented by the integration of such codes into a user friendly, robust multidisciplinary problem-solving environment.
One of the most interesting concepts that will be investigated through the current research activity is represented by the application of MBSE methodologies for the definition and development of a PSE tool. A lot of research initiatives are currently addressed to the evaluation of MBSE methodology as an advantageous approach for the definition of a modeling framework ([57]). Their main objective is represented by the creation of a high-level system model framework with which all the disciplines are directly connected. In this way all the information are centralized and shared through the same platform, reducing the problems related to data consistency and enhancing the collaboration between the people that are working on the same project. Interesting results have been obtained from some European research initiatives which are currently working on the definition of data structure and development of system modeling environments, basically following the model-based paradigm.
This MBSE methodology applied to modeling framework is actually implemented using various architectures and languages (UML, System Modeling Language – SysML, Domain Specific Language – DSL, etc.). Some of the developed frameworks are based on desktop application which show interesting performances on data processing but are not well suited for the integration with other tools within a distributed and collaborative environment (e.g. based on corporate network). One of the most promising solutions is represented by the definition of a web-based service that can be used to integrate different kind of resources in the same environment. The integration between modeling and analysis environments is a challenging research topic that involves a wide range of engineering issues. Nowadays the advancements of web-developing technologies allow implementing modeling architecture that can be directly interfaced with analysis resources. The benefits that can be obtained by such infrastructure concern mainly the capability to reduce time and costs of system development, ensuring also the consistency of shared data among various engineering domains.
Currently there is not a well defined framework that integrates both a modeling environment and a PSE workspace but in the last few years companies and organization have addressed a large amount of efforts in this direction. Commercial system modeling solutions provide different functionalities that enable the integration with networks and distributed environments, represented however by desktop applications that are often difficult to maintain and spread within system engineering domain. A web-platform shows many more benefits than drawbacks at the moment.
The learning curve related to such collaborative environment is in fact steeper than that associated to desktop applications since a web-based service is something with which everyone deals daily. The capability to choose the correct optimization strategies and solving methodology will be further improved through the connection of a PBE framework with a system modeling workspace. A clearer definition of the design variables and the integration with a PSE on the same platform will especially enhance the achievable results,

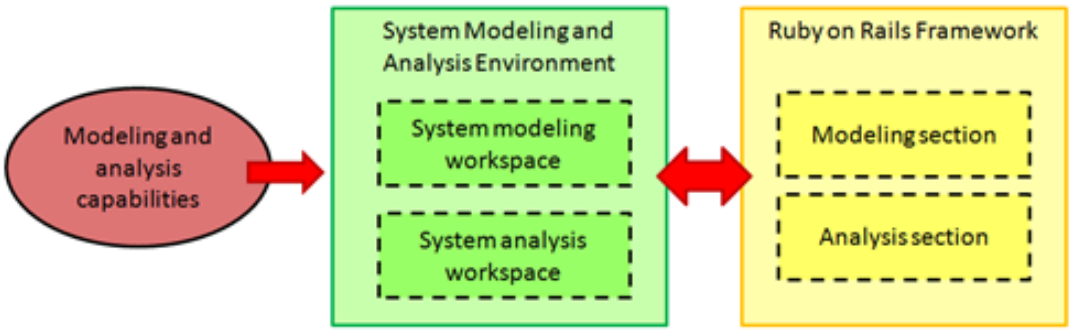

Figure 7.16: Overview of the conceptual infrastructure and related actual implementation.

allowing a better evaluation of system performances. The development of such platform can accelerate the design process, giving a considerable advantage with respect to other possible competitors. The reduction of current development time due to less project iterations is one of the main objectives of the proposed framework. The parallel design of system and operations is not yet fully implemented and used. Such an approach shows an innovative design methodology towards which some research initiatives are addressing huge efforts. The proposed methodology and related demonstration framework can be seen as an evolution of the current design procedures which are often not suited enough to handle increasingly complex systems.

This evolution includes a reinvented product life-cycle able to easily manage changes during each phase and also presenting social aspects to involve different actors in the project like designers, manufacturers, users and customers, reducing the time disposition for a problem solution or a simple communication up to real time, irrespective of the physical distance between the actors. Parallel development, i.e. system design performed concurrently with utilization definition, is to avoid over design or under design of the system that has to accomplish the assigned mission. In both cases this means both avoiding the introduction of additional development costs and reduced system utilization capabilities.

That lead also to a more comprehensive and effective trade-off analysis, easing the sharing of up-to-date controlled data and information. Another benefit of such infrastructure is represented by the cutting down of distances between actors, being closer to the customer's needs and expectations. Such integrated PSE will provide a better sharing and exploitation of the available resources, avoiding the slowdown due to an inefficient organization of the design and analysis infrastructure. A small overview of the proposed infrastructure (from an high level perspective) is provided in figure 7.16 where system modeling and analysis needs are used to identify the conceptual infrastructure for the proposed methodology. The same conceptual environment can then be implemented using different solutions to investigate and validate the proposed infrastructure. In the current work the developed meta-model and concepts have been evaluated through a web-based platform through Ruby on Rails (RoR) framework but alternative tools can also be considered. More details about such development environment and the reasons that have lead to such choice will be presented in the following.

The main focus of such research activity is in fact to investigate the model-based approach and related conceptual infrastructure, basically using one of the possible implementation solutions. The same result may probably be demonstrated with other development frameworks since there is no particular restraints on the actual means that can be used to explore and assess such methodology.

7.4.11 Proposed approach for the integration of MDO techniques

The integration of Multidisciplinary Design Optimization methodologies has been conceived mainly thinking about the possible uses and improvements in the context of system design. In particular the final purpose of such approach is identified with a clear understanding of how such an instrument can provide useful support and real benefits with respect to the traditional processes. A first analysis of the design

procedures, tools and people involved has represented the starting point for the definition of data model architecture. Such concept has been taken into account through all the development process to ensure that the objectives of the present work would be fulfilled.
Considering the integration of a multidisciplinary capabilities within a collaborative environment has laid the pathway to a conceptual definition for the relationships of the entities involved. First of all some considerations have been done on the final target and results that the framework is supposed to deliver. The possible solutions related to the integration of MDO techniques is strictly affected by way the interaction between users and framework capabilities works. The overall process has been conceived with emphasis on the final purpose, proposing an approach that is mainly tailored for a collaborative environment.
Different strategies have been considered to face the problem on the basis of the target services that are provided. The main scope is the definition of an architecture that allows the exchange and integration of models coming from different domains, paving the way for the set-up of multidisciplinary analyses. Multidisciplinary analyses more generally involve models coming from different modeling environments and tools. The investigation of overall performances is driven by the integration of such models among each other. In particular such situations are generally characterized by very non-linear problems where often the design variables domain lead to a complex response range for the generic output evaluated. The solving techniques used to manage such kind of problems are not the gradient-based ones since in these cases the risk to remain bound to a local minimum is quite high, reducing the possibility to identify an optimal solution. These types of problems are generally handled with non-gradient based methods (as for example heuristic genetic algorithms). This approach requires however a great number of function evaluations to properly explore the design variables space. Such amounts of simulations cannot be done in a short time if complex models are used but in this case the primary purpose would not be satisfied. The same approach is not limited a priori to simple models since the same architecture can also be used for more complex ones. The latter situation requires more time to obtain the desired results but it is not prevented and the choice between the two conceptual alternatives depends on the specific need. The modeling framework is basically implemented to support collaborative interactions between various users and models are directly managed through server. In the case the simulation complexities increase the models are generally referenced by the file-system of the related repositories and not directly stored on server side. Short time simulations are quite important in the context of a collaborative process where an iterative elaboration of the available resources allows users to draw their considerations or make their dimensioning computations.
Proper procedures to manage the available models are required to wrap the simulation functionalities and ensure the capabilities to partially automate their execution. This feature is directly related to the definition of a multidisciplinary analysis and the elaboration of the produced results. A simple conceptual representation about the considered architecture is reported in figure 7.17.

The implementation of the objects directly related to the analysis environment has been done following the conceptual data structure defined and assessed with the present work. The meta-model definition related to this aspect is mainly based on the formalization of three classes identified with the following names: “Analysis Item”, “Simulation Item” and “Analysis File”. In particular the relationships among these classes have been used to build the infrastructure with the final aim to pave the way to multidisciplinary evaluations. Ruby on Rails development platform and REST architecture philosophy have been used to code all the related associations as in the other sections of the modeling workspace. The links between such elements are conceptually represented in the figure 7.18.

The “Simulation Item” class has mainly been introduced to support a clear separation between the multidisciplinary analysis environment and the modeling one. In this way the design approach can proceeds in a more independent way with respect to the possible definition of system survey, avoiding the problems that can appear if these two environments refer to the same elements. The components properties can then be defined on the basis of the values provided directly by the user while the data computed through “Simulation Item” objects are available as support functionality. Such class in particular has been conceived to support the definition of the iteration cycles related to DAKOTA environment. Computational blocks can be created following the related attributes and associations that are available within the developed meta-model. The integration of simulation elements must be compliant with the exchange data

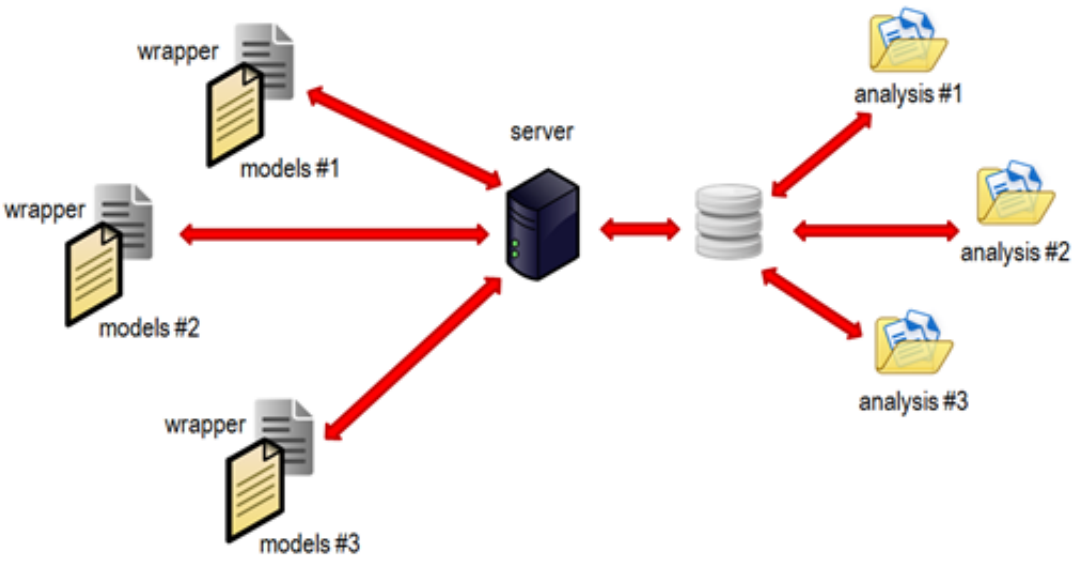

Figure 7.17: Conceptual representation about the considered architecture.

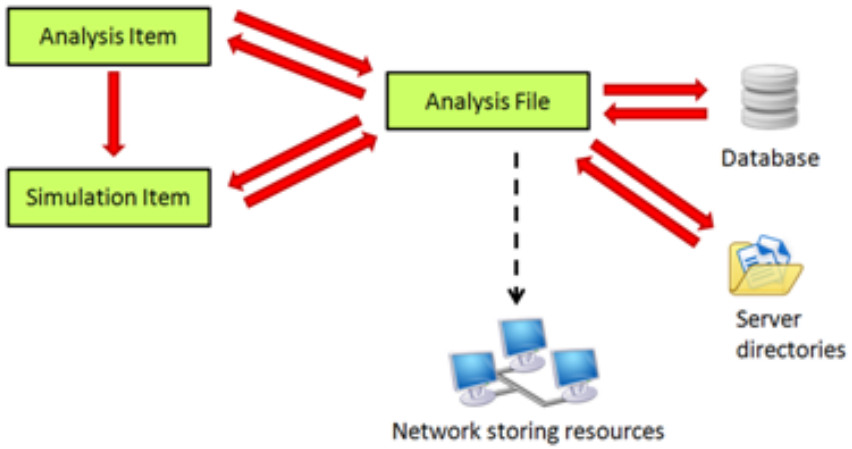

Figure 7.18: Example architecture for the considered approach.

Figure 7.19: Example implementation of python wrapper.

formats formulated for the current version of the developed framework. The convention refers mainly to the creation of the correct formats of the text files used to exchange the variables from one simulation to the other. The encapsulation of solving codes and simulations is implemented through Python wrappers that are customized on the basis of the contained elements. The so defined wrappers depend strictly on their contained simulation objects and if quite huge changes are introduced in the related solving code they must be properly updated before overall iteration take place (figure 7.19).

The correct integration of a "Simulation Item" element from a user viewpoint follow basically the following brief list of operations:

1. The simulation code is validated and proper working.
2. Input and output variables are identified.
3. A Python wrapper is properly is defined on the basis of the convention on input/output files.
4. The execution command of the wrapper, the files required for the simulation, the wrapper itself are upload and registered through the framework.
5. The related simulation capability is now ready for use.

The analysis section provides both a single run capability as also multiple ones once a simulation item is available. The single run execution allows to see the results related to a specific simulation item, providing a support for a particular evaluation. The results can also be analyzed through the post-processing capabilities provided by JavaScript codes available in the workspace. The multiple run execution refers to the surveys that can be created within the same environment and that generally involve other simulation items to build more complex multidisciplinary evaluations.
Surveys set up can be done through the functionalities provided by a separate section of the system modeling workspace where all the information defined in the current project can be used to drive the definition of the required analyses. In particular the developed user interface allows to define input and output of a specific analysis, loading also the simulation item currently available within the same project or provided by a resources library. The same section allows the definition of the relationships between the imported simulation blocks, generating the driver file required for the management of the single iteration by DAKOTA multidisciplinary analysis platform. The overall iterations cycle settings are managed in the same way, providing a set of different options and alternatives that lead the correct generation of the program main file. In this case the user interface has been conceived to drive the user to the creation of such file without the need to directly interface with the command line instructions, reducing the possibility to erroneously introduce wrong parameters. Such approach allows to better exploit the capability offered by DAKOTA, providing also a better understanding of the available solving techniques the choice of which widely affect the identification of the optimal solution. The results generated from the iterations cycle can be post-processed in the same environment through the available set of JavaScript codes for the management of such information. In this way the responses provided by the multidisciplinary platform can be properly elaborated by the user and taken into account to rightly guide the following phase. They provide only the

instruments that support the design and analysis processes but the final choice about the actions that must be done depends on the single user or a team of persons. The main purpose is in fact to clearly provide all the possible information with respect to the considered simulation scenario and system data. The potential change of system architecture or components properties depends however on the people involved in the project.
Data structure has been conceived also to foresee the potential integration with external computational resources, provided for example through server machines assigned to such a role for a specific model simulations. In this case the concept of simulation item is still valid and such observation shows the effectiveness of the conceptual architecture of the developed meta-model.
The integration with DAKOTA platform is not the only possible solution since this configuration has been chosen only to demonstrate the benefits of such model-based methodology while the attention is mainly focused on the validation of the developed meta-model (expressed through the related data structure), identification of the actual issues and investigation of the possible improvements. Other multidisciplinary design analysis platform can in fact potentially be considered as OpenMDAO framework for example.

7.4.12 Web application and networking

One of the most challenging research topics of the last few years are represented by the increasing integration of web-based technologies within the engineering design field. In particular a web-based application provides services that are quite difficult to obtain with the tradition approach. The development of a such an application has many advantages with respect to a desktop installed software also if this one covers a fundamental role. The industry knowledge is strictly influenced by the historical background bounded to the desktop application that have been used for several years in the design of system model. From the same point of view application installed on a certain platform (or more generally referring to a server machine that is however directly related to a desktop terminal) shows performances that are not so close to the ones obtainable with a web-based architecture. This last solution enhances however some other increasing benefits from the collaborative perspective. The share of information and data exchange is better managed through the implementation of an application referred to a server machine for the providing of the services. All the platform connected to the central repository can exploit this advantages of a common shared source of information without the need to install something to access such data. Networking is in this sense one of the most promising alternative for the management of system model data above all from a collaborative viewpoint.
The considered integration can also be seen in the context of Service-oriented Architecture (SOA) [89]. Generally such architecture represents a design pattern for the collection of different software modules. In particular the main target is represented by the generation of a single and large software application based on the functionalities offered by various services (other software). This architecture is mainly conceived to ease the cooperation between different computers connected over the same network, improving the capabilities offered by the overall system. The orchestration of such services is one of the most important features for the correct functioning as also a well-defined communication protocols. The benefits of SOA can be mainly identified in the simultaneous use and in the easy mutual data exchange between programs of different vendors often running on different platform (Windows, Unix, Solaris, etc…). All these capabilities are provided without additional programming and with a more consistent information exchange. The federation of resources is one of the most challenging aspects of the current research topics since it potential offers some interesting feature above all in the context of a collaborative environment. This approach requires however a well-established data flow to a federated database system. Another attractive element related to SOA concept is represented by the fact that it is principally based on object oriented design since each service can be seen as a discrete piece of code, providing characteristic functions and methods. Such feature underlines also the reusability of the code itself by changing only the manner the individual service interoperate with the other ones within a certain main application. The clients access to the services provided by the complete application is managed exploiting the well-defined interfaces such for example XML. Although such format is quite widespread for the interfacing of SOA services, JSON is increasing its presence over such architecture.

Standards compliance, service identification, reusability and modularity are some of the most important elements that ensure a correct behavior for the functionalities provided.
Web services can be implemented following the main feature related to the previously introduced concepts. In this case two main roles can be identified and are represented by Service provider and Service consumer (for example the client). In particular such implementations can be defined generally as Web-oriented architecture (WOA) and they basically extend service-oriented architecture to web-based applications. Generally they are also considered as a sort of light-weight version of SOA and are mainly aimed to increasing the interactions between browser (client) and server by REST or POX technologies for example. The implementation of SOA often follows the already defined web standards (for example SOAP) that have gained a quite broad acceptance over industry but such architecture can also be defined using any other service-based technology. In the following list are reported some of the possible alternatives:

- SOAP, RPC
- REST
- DCOM
- CORBA
- Web services
- DDS
- Java RMI
- WCF
- Apache Thrift

7.4.13 Web application integration alternatives

The management of information through web application can be realized through proper developed scripting language interfaces. For example Python language offers a series of modules and functions that can be used to access and exchange data over the network. For example **XML-RPC** is a remote procedure calling using HTTP as the transport and XML as the encoding. XML-RPC is designed to be as simple as possible, while allowing complex data structures to be transmitted, processed and returned.
Other similar web services can be provided using other modules like **JSON-RPC**, which is quitter similar to the previous one but has been based on the communication through JSON file extension.
Another way to integrate a web service for the management of Python functionalities is represented by the generation of a proper **Django** framework. In this way the front end can be managed through Ruby on Rails while some of the scientific libraries already available in Python can be accessed across the network. In particular this support application can be considered only for the exploitation of certain scientific computation but not for the management of the other web feature as user access, data storage, etc… which are instead provided directly with RoR application. Django framework can be used in this way to build the overall Python application since it provides some interesting features ready to use while the other considered alternatives require a lower implementation level. This characteristic lead to a more synthetic implementation of the required application but at the same time is not so easily managed from a development point of view. As previously introduced many different web development frameworks can be chosen for the integration of the Python libraries (like for example Zope2, web32py or TurboGears) but Django shows a well-documented and supported set of resources.
The integration of RoR web application with a Django service allows to exploit the scientific libraries already defined and validated in Python. In this manner a more dynamic interface can be used to integrate some of the utilities available within Python environment without the need call Python script from Ruby. This approach is based mainly on XML and JSON data exchange between the web services, using in particular

the advantages of AJAX paradigm to interface the elements of the proposed framework. Data can be sent to a Python web application to be properly processed and then returned to the calling function through JSON format. The same method can also be used to render XML data over HTTP requests.
This approach allows also for a better management of updates and maintenance activities since the application runs on the same machine. The data requested are instead distributed to different clients using the web browser on various platforms. For the same reason the Python application not necessarily musty be hosted on the same machine of the RoR application. It can be placed in fact on a different location available on the network.

7.5 Expected results, their significance and application

A web-oriented architecture can widely improve the collaborative process when the project involves a large number of engineering domains. A web service solution can be developed to support multidisciplinary analyses, providing all the elements required for the management of the available computational resources. A set of useful utilities will be accessed by the single user directly through the web browser, paving the way to a more effective management of the shared resources. The same web interface will also be used to adapt PSE to Information Power Grid (IPG) environment for MDA applications.
The main focus of the proposed research activity will be addressed to the assessment of a model-based approach for the management of both modeling infrastructure and computational resources. It is expected that the developed framework will help the definition of multidisciplinary analyses through the use of a web-based interface directly linked with a system modeling environment. In this way it will be possible to reduce the problems related to the exchange of inconsistent information, allowing also building a more structured evaluation of system performances. The framework delivered with this activity will allow accelerating product design and manufacturing process, to supporting multi-domain systems engineering, simulation-based engineering, and knowledge management, besides the current design approach. The evaluation of the proposed approach on a reference case will be used to validate the applicability and usability of the developed framework. The comparison between the results obtained on the same case but with both the traditional approach and the proposed one will be used to identify actual benefits and drawbacks.
The developed framework will show how the model-based infrastructure can widely reduce the time and costs of product design. A better management of the available resources concurrently with a proper connection with a modeling environment can help to overcome those inconsistencies that often slow down a smooth design process. The developed framework will provide a set of functionalities that will be divided basically in two workspaces. A modeling workspace will be mainly used to define all the features and properties that will drive system development. These capabilities will be used by system engineers to define product architecture with also the possibility to detail design and components as the project proceeds to more advanced phases. In this way system information can also represent a reference baseline for all the involved engineering domains, paving the base for the correct integration of simulation models. The other workspace will be mainly used to support all the activities directly related to the management of computational resources and multidisciplinary environments. The synergetic integration of both these workspaces in the same PSE will widely improve design process since all the data will be available in the same distributed environments among all the actors.
The methodology will impact the time needed for the interfacing with customer and supplier, easing the requirements elicitation phase and potential changes and discussions due to misunderstandings. The time related to the passage from design to operations phase (typically managed by different teams) will also be positively affected. Another activity that will be improved from a time saving viewpoint is represented by the reconfiguration of items during operations and feedback on new design (i.e. easier definition of new requirements for improved space application).
The proposed methodology is basically not limited to large-scale aerospace problems but can also be applied to other complex applications such as bio-engineering, shipbuilding, civil-engineering or automotive ones. Different engineering fields can basically be approached in the same way, providing the basis for a

common design methodology. In this way the use of different development platform that often limits the data exchange between disciplines can be reduced, enhancing the creativity and innovation. The application of the proposed MBSE methodology and related infrastructure can also be applied seamlessly the design of other complex aerospace systems as Unmanned Aerial Vehicle (UAV) for example.
The same environment can provide useful utilities to properly support the decision making process, avoiding the problems and misunderstandings that often arise from information inconsistency.
A seamless integration between design and analysis can be improved starting from the implemented framework, paving the way for future developments addressed to the improvement of product data exchange. System realization process can also be understood in a more effective manner with the main purpose to avoid differences between the designed system and the produced one. Such a model-based framework can also be implemented to store and re-use design history from previous projects, improving the overall effectiveness of resources utilization. The developed environment can be used to support the determination of the impact of decisions with respect not only to aerospace applications but also to other engineering domains. In fact the proposed approach and related framework can also be easily applied to the design processes of other fields. The formalization of the developed infrastructure can also promote a continuous learning, enhancing the capability to design and manage complex systems once a common modeling and analysis approach has been defined. In this way the information infrastructure can be improved, reducing or ideally eliminating the possibility that incorrect data are exchanged across the actors involved in a project.
The integration between different analysis tools is another interesting feature that can be widely improved through the definition of a common platform that works as a reference point for all the disciplines involved in a project. The identification of globally optimized designs can then be pursued also with the support of utilities dedicated to the management of complexity and risk in the same environment. Another benefit is represented by the creation of a common platform for the definition of performances evaluation methods shared in the same collaborative framework.
The developed web-based application can also be used to improve the communication of design specifications to remote sites and companies not necessarily limited to aerospace field.
The formalized integration of large-scale systems allows harmonizing all the involved domains, with the possibility to partially automate the conversion of exchanged data through the definition of appropriate interfaces (thanks to the formal definition of data structure). This capability is mainly due to the agreed data structure that is shared among the involved disciplines and allows also reducing the error-prone and often time-consuming process of information conversion.
The formalization of the concepts related to simulation codes and design methods can help to trace and store important information generated internally by a specific company or organization. In this way it is possible to improve the return of knowledge from a certain project, enhancing the reusability and modularity of already implemented features and resources. It is not uncommon that some working codes and scripts implemented by an individual user is not shared with other persons since it is generally stored on local machine and it is customized for specific purposes. The same functionality is not uncommon that is re-implemented from scratch in another project from another person, consuming resources (time and efforts) on something that can be potentially avoided with a shared infrastructure for the exchange of such information. In this way people can use already developed equations/functions or codes addressing their work on another activities, improving the overall design process. In the opposite case all the achieved capabilities are easily lost and the design process proceeds as always in a not proper effective manner.
The application of the same developed environment can be easily customized with the final purpose to support not only aerospace industry but also other engineering fields. A better tracking of design changes and related costs will reduce the lead time in product modification, needed for example to meet new business demands. The improvement of system compliance with respect to customer needs is in fact a common concern to various industrial domains.

Chapter 8

Reference Case

In the following sections the results achieved through the implemented infrastructure will be provided and described. In particular a reference case has been considered to evaluated the proposed framework with respect to the integration of multidisciplinary design and analysis techniques. The problem is introduced in the first part, reporting the main features and issues that characterize such kind of analyses. In this section the formalization of the problem is specified, detailing the considered design variables, objective functions and project constraints. The multi-objective reference case is then resolved exploiting the techniques provided by DAKOTA platform and integrated in the same modeling infrastructure. The proposed approach has been implemented following the principles of model-based philosophy, trying to set up the target capabilities with an object-oriented structure as much as possible. The main idea is to assess the feasibility to clearly separate the problem formulation from the actual solving methods and routines, enhancing the management of complex systems. All information needed to set up a multidisciplinary design analysis are defined within the system modeling infrastructure and are used to implement specific surveys. In this way the same approach can be used for a wide range of engineering problems, improving the reusability of such an infrastructure and avoiding the implementation of a solution that can be used only for specific situations. In these case in fact the maintainability of such tools/models becomes difficult to ensure since they are often linked to a restricted group of people. The results generated from the solving algorithms are then reported in the final part of this chapter.
It is important to underline that the surveys considered for the analyzed reference case have been done to show the feasibility to connect system models and analysis environments within the proposed infrastructure. The assessment of the actual solving techniques as well as the implementation of alternative optimization techniques is in fact not the primary objective of the current work. The development of solving methods can however be pursued within the same platform in future works. The code accessibility provided by an open-source platform ensures inn fact the possibility to directly implement optimization algorithms to manage complex non-linear problems as in the case of aerospace products.

8.1 Introduction

The considered reference case has been selected from a range of possible design and analysis alternatives among different space applications. In particular a space mission scenario involving different engineering domains has been chosen to show how the proposed infrastructure can be applied for the management of multidisciplinary surveys. The space system under evaluation is represented by a human rated vehicle designed to reach as support spacecraft of another human outpost, located in the Lagrangian point L2 of the system Earth-Moon. The vehicle has been conceived to supply the system already present in the target position with support units or expendables materials for example. The Lagrangian point L2 represents an interesting position for space applications since it offers some advantages for the achievement of specific objectives. Orbits around Lagrangian points offer in fact unique advantages that have made them a good solution for performing specific spacecraft missions. Earth–Moon L2 is a good position for the location of a communications satellite covering the Moon's far side. At the same time an Earth–Moon L2 point would be also a good location for a propellant depot as part of the proposed depot-based space trans-

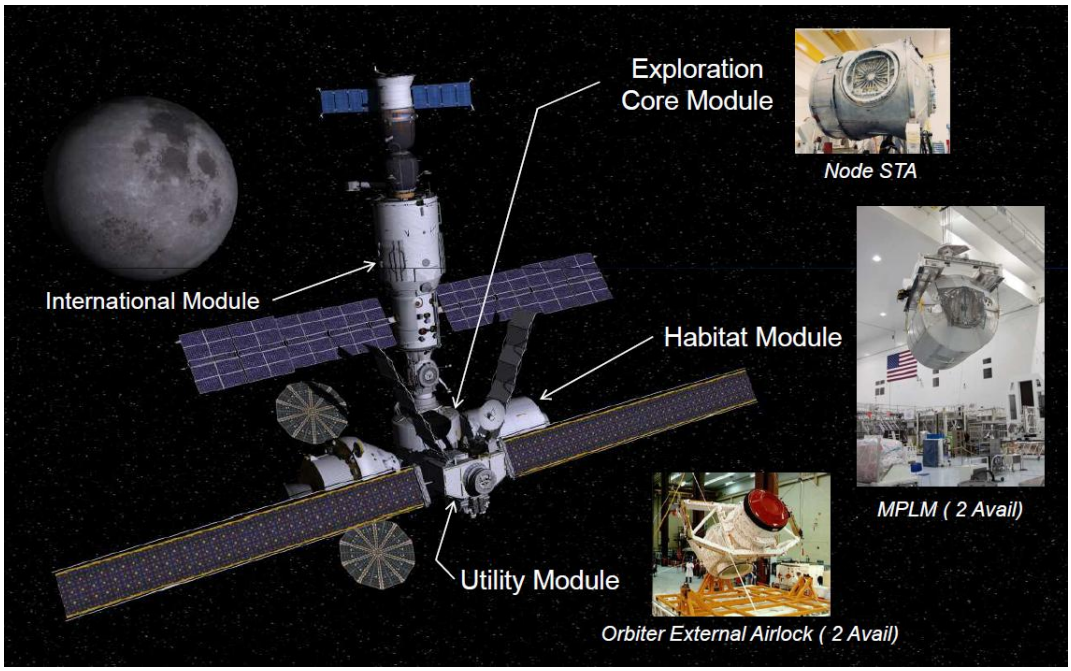

Figure 8.1: Conceptual representation of the project Exploration Gateway Platform [97].

portation infrastructure. Missions like ARTEMIS (which is a mission extension of THEMIS [95]) have been developed across the years to investigate and potentially exploit the benefits coming from the positioning of spacecraft in the Lagrangian points, considering both the Sun-Earth system as well as the Earth-Moon one. Currently different missions involving the Lagrangian points have been proposed for the next future and various surveys have been done for the preliminary phases of design.
An interesting mission is represented for example by the *Exploration Gateway Platform* [96]. This project has been proposed by Boeing in December 2011 to reduce the cost of Moon, Near Earth Asteroids (NEAs), or Mars missions by using components already designed to construct a refueling landmark and servicing station located at one of the Earth–Moon Lagrange points, L1 or L2. Cost savings can be achieved through already developed and deployed elements that can be reused for multiple missions such as a launch platform for deep space exploration, robotic relay station for moon rovers, telescope servicing and a deep space practice platform located outside the Earth's protective radiation belts. In this case the overall platform or related spacecrafts would be constructed at International Space Station (ISS) for testing before being placed to L1 or L2 via electric or chemical propulsion rockets. A conceptual representation of the considered platform is provided in figure 8.1.

8.2 Problem description

This reference case have been selected since it shows some of the main issues related to the definition and management of multidisciplinary analyses. The design of space mission requires often the correct understanding of all the variables that are shared among the different disciplines and actors. The proposed infrastructure has been conceived to manage such kind of information from the preliminary phases to the more advanced ones. The actual solution of the involved design parameters with respect to the final objectives strictly depends on the available resources and related needs for the specific project. For this reason the implemented infrastructure provides common capabilities that can be used among different programs since they are not bounded to a specific system model. According to such considerations the system engineers can use such infrastructure to support the decision making process in the preliminary phases as well as the more advanced ones. In the first case simplified models of different disciplines can be connected together to implement system model simulations, paving the way for a more effective way of design space exploration for example. In this case simulation results of multidisciplinary analyses can be achieved in a relatively short time (since the performances are directly affected by the available com-

putational resources). Similar analyses can be executed in the second case on more detailed models and scenarios. The same approach and related platform can in fact be used also in this situation but the surveys will be addressed towards more specific analyses with a reduced range for the design variables.

8.2.1 Main issues

The main issues directly related to the problem under consideration are represented by the effective set-up of different simulation environments and codes. The same issues can also be found in other similar design problems and for this reason they are discussed in this section. Such issues are approached through the concepts of *Function* and *Function Model* within the proposed approach since these objects allow the connection with the information contained in the topological elements. In particular the links with externally implemented solving codes must be properly managed to avoid unpleasant situations where codes not completely known run on server machine. This situation must be properly taken into account to ensure the persistence and correctness of the stored data and avoid uncontrolled process on server side (this situation can potentially represent in fact a dangerous threat). The possible solution for the management of such kind of simulation routines is represented by the definition of a properly monitored registration process for the related resources. Also in this way the overall process cannot be directly controlled since the main problems regards the possibility that the users can have to upload externally developed codes on their own (such parametrical models can be used to define the core capabilities of the functions themselves). In this case it is necessary to monitor the overall process when such codes are uploaded and properly integrated within the infrastructure. They are directly stored in the server machine and are available to other users of the web-platform once the "registration" process ends. This kind of problems will always be present each time we try to let users to upload and define the codes that will be integrated within the *Function Model*.
The other possible solution is represented by the fact that the simulation codes are directly defined within the system modeling environments, choosing a common modeling language as Modelica for example. In this manner the codes are stored as text in the *Function Model* but in this case they can always be processed to identify possible dangerous commands for the server machine. The uploaded code can in fact be parsed to identify possible threats and rise a warning of the code itself. Modeling languages like Modelica can be used in this way to implement simulations belonging to different domains (thanks to the multi-domain capability of Modelica itself). With respect to the previous considered alternative the codes can always be directly controlled and accessed to verify that they are not dangerous when executed on the server machine. The disadvantages of such solution is mainly related to the fact that this modeling language (in analogous way another language that must be learned) requires a training period if the user does not know it. In this case already developed codes must be converted and re-validated to ensure the correctness of the generated results with respect to the already available ones. The codes themselves can be exchanged, updated and maintained with minor efforts with respect to codes previously developed once the same capabilities have been developed and deployed.

8.2.2 Analysis of the problem

The main aim of the survey supported with the proposed infrastructure is represented by the investigation and evaluation of the performances about the possible mission solutions. In particular a set of design variables is defined on the basis of the data available from the topological design. In particular the design variables considered for the current problem regard for example the choice of the launcher, the geometrical properties of the spacecraft, electrical power system alternatives, etc. A more detailed description of the design variables is provided in the section related to the explicit formulation of the problem. A set of constraints is then considered to bound the feasibility of the solutions selected, avoiding the possibility to identify configurations not actually possible. Such constraints mainly deal with the structural feasibility of the primary structure of the system, the minimum temperature experienced within the spacecraft as provided by the thermal simulation, etc. More details about constraints definition are however provided in the next section. The main objective of the considered mission can be summarized with the maximiza-

tion of the available mass in the final orbit position but at the same time other factors must also be taken into account. The cost of the whole mission represents for example a limiting parameter that affects the final choice of the spacecraft configuration (other objectives are also taken into account and are widely described in the following section). The considered case shows the characteristics of a multi-objective design problem and the available information must be properly managed to identify the most suited system. A strong non-linearity is basically highlighted by the characteristics of the simulations involved and the presence of a discrete space for some of the design variables make the problems not so easy to approach. The identification of the optimal solutions is difficult to achieve in a straightforward way in this kind of problems. Different non-gradient based algorithms and routines can help to pursue such search and a properly defined system model infrastructure provides the basis for a consistent representation of the required data.

8.2.3 Description of the involved disciplines

The performances evaluation of the whole system requires the execution of different domain-specific codes that allow the computation of the desired quantities. The values needed to get an evaluation of the system capabilities are in fact widespread over different engineering domains and the output from one simulation may be required for example as the input for another one. The data exchange between the simulation blocks (conceptually identified with the *Function Models*) is defined through the correct implementation of the data-flow. This process can be supported through a graphical user interface available within the framework itself.
The main disciplines involved in this preliminary design are represented by the mission analysis domain, structural domain, thermal domain and electrical domain. The computations related to the mission analysis domain are mainly related to the evaluation of the trajectory characteristics, delta-V and payload capability (directly related to the choice of the launcher). The structural domain is instead involved in the computation of the main stresses experienced by the structure on the basis of the launch loads (also related to the main characteristics of the launcher) or pressurization loads (due to the fact that the mission is constrained to a human rated mission) for example, taking into account the main geometrical properties that have been considered in the set of the design variables. In this case such information are used to understand if the primary structure is able to withstand the external loads on the basis of the specific material properties. The evaluation of the thermal response of the spacecraft with respect to the external environment and on the basis of the geometrical quantities is attributable to the thermal domain. In this case the obtained information can be used for example to assess if the internal temperatures have dropped below the limit values for the survival or comfort of the crew (taking always under consideration the development of a human rated mission). In the end the electrical domain involves all the computations needed for example to evaluate the required solar arrays extension as well as the estimation of the mass allocated for the energy storage system. In this case the information gathered within the system model (represented for example by the main characteristics of the components that can be used to assemble the final configuration) is used to generate the required output. Different solar cells technologies and battery types can in fact be considered and part of system performances are directly affected by these choices as well as for the other design variables that involve the other domains.

8.3 Problem formalization

The problem approached in the current section can be explicitly formulated to show the main features of the whole analysis. In particular the various simulation models (directly connected with the *Function Models*) are used to generate the required data and the information are exchanged among each models to evaluate the performances of the system. The implemented framework currently allows the management of the Individual Discipline Feasible (IDF) architectures and the considered reference case consistently follows such structure. The approached problem reflects in fact the main features of an IDF pattern and it has been considered to assess the capability to manage such type of analysis.

8.3.1 Simulation models

The overall simulation of the system is based on different simulation models that belong to different domains. Each simulation model has its own input and the generated output are available for the other ones, the data flow among the involved simulation modules must be defined during survey set-up.
The main features of the mission trajectory are computed taking into account a simplified transfer orbit from the initial circular orbit (around the Earth) to the final orbit near the Moon. The transfer orbit is an Hohmann transfer trajectory, assuming that such estimation is quite suited for the purpose of the survey. The input of such module are represented by the launcher type (parameter that directly affects the payload capability for the starting orbit), the target altitude of the initial circular orbit and specific impulse for the kick-off motors. In this case it has been assumed that the kick-off motors used to enter the transfer orbit to the Moon are provided as external service to the designed spacecraft. The same propulsive subsystem is also used to manage the positioning in the final location, assuming that the most part of such servicing does not belong to the main spacecraft. For this reason a specific impulse is associated to such block. The same module takes into consideration the variability associated to a set of possible launch sites, also if this input does not directly affect the main performances of the whole system, excluding the time to launch with respect to an initial relative position between the Earth and the Moon. The main output are represented by the final mass in the target position, the time of flight and the cost associate to the launcher (derived from the configuration selected for the simulation). Payload capabilities, expressed as the payload mass function of the target altitude, are obtained from the data sheets of the launchers (interpolated from the experimental data).
The computations related to the structural module allow to estimate the main stresses that characterize the primary structure. In particular the pressurization loads and maximum accelerations in the axial direction are used to compute both the hoop pressure stresses as well as the ultimate compressive ones. The structure response is calculated on the basis of the geometrical characteristics of the structure itself (external radius, thickness and length of the primary structure). The simulation block take into consideration the density and the elastic module of the possible materials used to implement the structure (a set of aluminum-alloys are considered as input variables). A set of safety and weight factors is also introduced and it is assumed as a project data. Such information are used to estimate the response of the structure considering the proper margins. structural stability. The mass of the primary structure and the available volume within the spacecraft are generated from this simulation block.
The estimation of the thermal response is obtained by the parametrical model associated with the thermal function. In particular the corresponding model is defined to assess the system behavior with respect to the mission scenario. In particular the thermal simulation allows to estimate the product response during the transfer orbit from the Earth to the Moon (the simulation time is computed according to the transfer orbit available from the mission analysis). In this case some of the material properties, like aluminum-alloy density or thermal conductivity, are used to set-up the thermal network used to solve the problem. In this case the geometrical data related to the whole system (external radius, length and thickness) allow to define the thermal capacitances of the thermal nodes contained in the network (Matlab©script). The overall network is solved taking into account the thermal resistances among nodes and computing the thermal fluxes between the involved elements. In this way the time evolution of the temperatures is obtained through numerical estimation. This information is then processed to assess the minimum temperatures within the module during the transfer orbit. The thermal control is ensured through the assumption of the presence of heaters components within the spacecraft. It is also assumed an ON-OFF thermal control law that is activated or not on the basis of the current temperature of a node assumed as a reference sensor. The temperature evolution is estimated on the overall simulation of the transfer orbit (computed in the mission analysis block) and taking into account also the maximum power available for the heaters (provided as an input parameters). After the simulation the minimum temperature registered within the spacecraft and the total energy consumption are returned as output.
The main features related to the Electrical Power Subsystem (EPS) are computed in relation to the available information from the project data. The power requirement takes into account the baseline demand, considering also the peak power levels and introducing also a margin for the related computations. It is

assumed that the primary source power is represented by the solar energy. The cells physical characteristics and properties are used to estimate both the mass and surface extension of the solar arrays. A set of possible alternatives for the battery types is also considered and the related information is used for the dimensioning of the primary storage system. Other data as the Depth Of Discharge (DOD), battery specific energy and density are taken into account. The time during which the batteries are able to supply the required power is constrained as a project data as well as the time allocated for the re-charging of battery-pack.

8.3.2 Design variables

The design variables considered for the current case are briefly discussed in this section and are used as input variables for the simulation models described above.
A design variable is represented by the launcher type. Three main classes have been taken into account, considering also the associated configurations. In this case 33 possible solutions have been included in the analysis and the related variables has been expressed as an integer number. The three reference classes of launchers are: Atlas 5, Delta 4M and Ariane 5ES (Evolution Storable).
The target altitude of the circular orbit for the initial positioning of the spacecraft and propulsion-service module is another design variable. In this case a range between 400 and 1400 Km has been chosen as the possible starting orbit for the following transfer to the Moon. In this case the design variable belongs to the continuous domain. The evaluation of payload capability is based on the data-sheets available from launchers catalogue (as well as from the literature) as the one visible in figure 8.2.

The longitude of the launch site is provided as a design variable but does not have particular influence on the overall system performances for the analysis under consideration (their presence is needed for the computations related to the mission analysis and are introduced to verify their influence through sensitivity analysis; their actual importance with respect to the other variables is lower since not all the launchers can be freely associated to a launch site and in the current case such possibilities are not managed through constraints; they are mainly introduced to investigate the capability to manage such kind of data). The longitudes of the launch sites are provided as a set of discrete real values.
The main geometrical characteristics of the spacecraft are managed as design variables and can be summarized with external diameter, overall length and thickness of the external wall (in this case averaging the thickness of the primary and secondary structure as well as the thermal protection cover for example). The spacecraft shape is assumed to be cylindrical. The external diameter is formulated as a continuous design variable in the range from 4.2 to 4.6 meters. The overall length is also modeled as a continuous design variable in the range from 5.5 to 6 meters. In the end the wall overall thickness is modeled as a continuous design variable in the range between 0.1 and 0.15 meters.

The specific impulse of the kick-motors for the transfer-orbit insertion and final positioning is modeled as a continuous design variable in the range between 200 and 300 seconds.
The overall maximum power allocated to the heating elements within the spacecraft is expressed as a continuous design variable that can take the values from 500 to 2000 watt.
The aluminum-alloy that can be selected for the primary structure is managed as a design variable expressed as an integer number. In the current survey three different types have been considered: Aluminum 2219-T851 1'' plate, Aluminum 6061-T6 sheet and Aluminum 7075-T73 sheet.

The solar arrays can be basically implemented using different types of cells technologies and the related variable has been defined with an integer number. In this analysis four typologies of solar cells have been considered: Silicon, GaAs dual-junction, GaInP dual-junction and thin films.
The last design variable considered in the current study is represented by the battery type used for the primary storage system and it has been modeled as an integer number. In particular the different battery solution considered are: Nichel Cadmio, Nichel IPV (Individual Pressure Vessel), Nichel CPV (Common Pressure Vessel), Nichel Metal-Hydrate and Ion Lithium.
The previous design variables, once defined in the modeling environment, can be directly connected with the simulation models through the survey set-up, during which the overall data-flow is defined. Such information are then used to compute the output quantities through which both the objective functions and

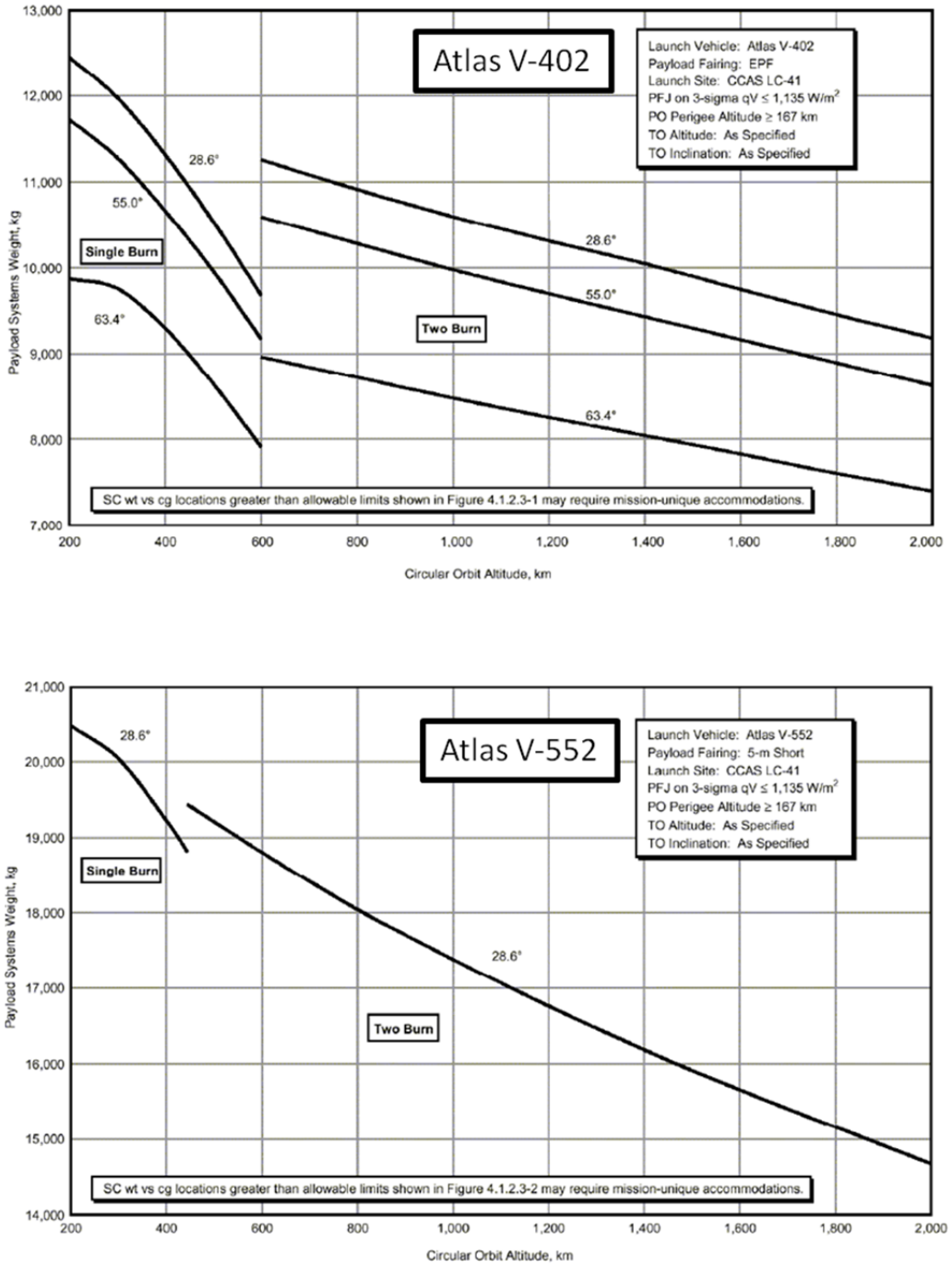

Figure 8.2: Example of payload capability expressing the mass as function of the altitude.

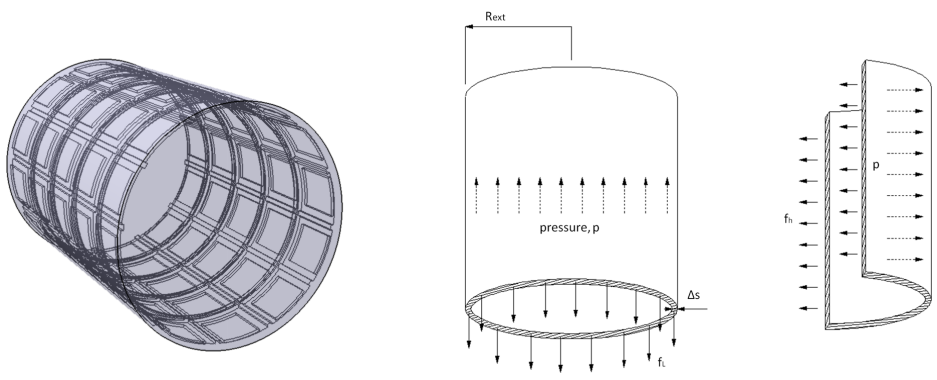

Figure 8.3: Simplified representation of the primary structure considered in the reference case.

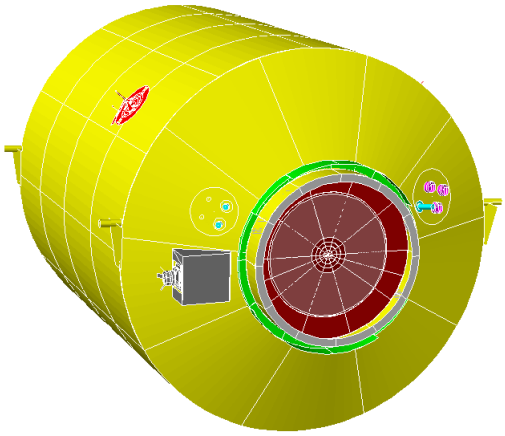

Figure 8.4: Simplified representation of the thermal model considered in the reference case.

constraints are defined.

8.3.3 Objective functions

The objective functions evaluated in the current survey are basically represented by: an estimation of the cost relate to the launcher, the mass in the final orbit (excluding the mass of the primary structure and EPS), the available internal volume of the spacecraft (excluding the volume occupied by the energy storage system) and the total energy consumption related to the thermal control system. In particular the purpose of the survey is the minimization of the cost, the maximization of the available mass in the final orbit, the maximization of the available volume and the minimization of the consumed energy.
The mapping of such variables with the quantities to be optimized within the current analysis is defined during the survey set-up. The related quantities are connected with the simulation items and are used from the external solving platform to drive the selection for the updated design variables.

8.3.4 Constraints

After the definition of the design variables and objective functions the following phase regards the formalization of the problem constraints. In this case there are both variables constrained by the project data (not directly involved in the optimization process) as well as other quantities that must be properly constrained to avoid unfeasible configurations. In particular the mass of the primary structure and EPS must be lower than the overall mass available in the final orbit. Such condition must be formalized since the related masses are computed by different simulation models and the feasibility of the selected design variables is not directly ensured. The minimum temperature registered within the spacecraft (during the transfer orbit to the Moon) must be consistent with the human-rated characteristics of the mission. Other constraints have been introduced to take into account for the material resistance with respect to the actual stresses of the structure as well as the limit associated to the elastic buckling load.

8.3.5 Solving methods

The multidisciplinary optimization platform used to investigate system performances is Dakota and the associated capabilities have been integrated within the modeling environment. The reference case approached with the proposed infrastructure is then characterized by 11 design variables, 4 objective functions and 4 inequality constraints. The reference case considered is basically a multi-objective optimization problem. There are three possible alternatives for multi-objective optimization in Dakota. The first is MOGA (Multi Objective Genetic Algorithm), the second is the Pareto-set strategy while the third is a weighting factor approach for multi-objective reduction, in which a composite objective function is built from a set of single objective functions using a user-specified set of weighting factors. The latter two solutions work

with all of the individual algorithm based based on single objective optimization. In these cases in fact the multi-objective optimization is approached through the "transformation" of a multi-objective problem to a single objective one (in this manner than it is possible to use all the methods planned for the solving of single objective optimization). In the future Dakota will consider also the integration of multi-objective response data transformations for goal programming, normal boundary intersection, etc.
The non-linearity of the system response with respect to the input design variables led to the choice of derivative-free global methods. They can in fact be applied to problems were gradient computations are too expensive or unreliable. In these cases the gradient-free methods are often go-to methods when the problems may be nonsmooth, multimodal, or poorly behaved. It is important to take into account, however, that they show much slower convergence rates for finding an optimal point, and for this reason, tend to be much more computationally demanding with respect to gradient-based methods.
MOGA belongs to the Evolutionary Algorithms (EA) class, available in Dakota (the other EA methods are SOGA and coliny EA) and it is basically defined following the Darwin's theory of survival of the fittest as the other EA algorithms. They generally start with a randomly chosen population of design points (from the ranges defined for each parameter space), and the values of the design variables are used to form a "genetic string". Such string does the same job of DNA in a biological system, uniquely identifying each design point in the population (a set of design variables is uniquely associated to a specific sequence). The EA manage a sequence of generations, where the best design solutions among the population are considered to be the most suited and are allowed to survive and reproduce for the following generation. The EA basically implements the evolutionary process through the mathematical analogs of processes such as natural selection, breeding, and mutation. In this way the EA try to find a design point (or a class of design points) that minimizes the objective function. A more detailed description of such optimization techniques are available from different resources as [99] or [100] for example.
MOGA is based on the idea that as the population evolves in a GA, design points that are non-dominated are selected to remain in the population. In particular it has separate fitness assessment and selection operators called the *"domination count"* fitness assessor and *"below limit"* selector respectively. This approach of selection works especially well on multi-objective problems because it has been specifically designed to avoid problems with aggregating and scaling objective function values and transforming them into a single objective. The fitness assessor works by ranking population members such that their resulting fitness is a function of the number of other point that dominate them. The below limit selector then select design solutions by considering the related fitness. If the fitness of a design is above a certain level, which in this case corresponds to a design being dominated by more than a specific number of other configurations, then it is discarded. Otherwise it is kept and identified to go to the next generation. The one feature is that this selector will require that a minimum number of selections must be done. The shrinkage percentage is correlated instead to the minimum number of selections that will take place if enough design points are available. It is defined as a percentage of the population size that must proceed to the next generation. To allow such approach, the below limit selector makes all the selections it would make anyway and if that is not enough, it relaxes its boundaries and makes selections from the remaining points. It continues to do this until it has selected enough designs. The MOGA method has however many other important features and a more detailed description of the involved parameters is available in [101].

8.3.6 Explicit formulation

On the basis of the previous descriptions regarding the design variables, objective functions and constraints the problem can be explicitly formulated as in the following lines.

$$\begin{aligned}
&\text{minimize:} && f_0(\vec{x}, \vec{y}(\vec{x})) && \\
&\text{with respect to:} && \vec{x} && \\
&\text{subject to:} && c_i(\vec{x}, \vec{y}(\vec{x})) \geq 0 && \text{for } i = 1, \ldots, 4
\end{aligned}$$

with the following notation:

$$f_0(\vec{x}, \vec{y}(\vec{x})) = \sum_{i=1}^{4} \alpha_i \cdot f_i(\vec{x}, \vec{y}(\vec{x}))$$

$$\vec{x} = (x_1, x_2, x_3, \ldots, x_{10}, x_{11})$$

Design parameters:

- Altitude of the initial orbit: x_1, continuous design variable, $400 \leq x_1 \leq 1400$ Km.
- Specific impulse: x_2, continuous design variable, $200 \leq x_2 \leq 300$ seconds.
- External diameter of the spacecraft: x_3, continuous design variable, $4.2 \leq x_3 \leq 4.6$ meters.
- Overall length of the spacecraft: x_4, continuous design variable, $5.5 \leq x_4 \leq 6$ meters.
- Average thickness of the spacecraft wall: x_5, continuous design variable, $0.1 \leq x_5 \leq 0.15$ meters.
- Maximum power available for thermal control: x_6, continuous design variable, $500 \leq x_6 \leq 2000$ watt.
- Launcher type: x_7, integer design variable, $0 \leq x_7 \leq 33$.
- Aluminum type: x_8, integer design variable, $0 \leq x_8 \leq 2$.
- Cells type: x_9, integer design variable, $0 \leq x_9 \leq 3$.
- Battery type: x_{10}, integer design variable, $0 \leq x_{10} \leq 5$.
- Longitude of the launch site: x_{11}, set of real values (enumeration), $x_{11} \in (-1.412, -2.105, -0.915)$ radians. The launch site corresponds approximately to Kourou Guiana Space Center (-0.915 radians), Vandenberg Air Force Base (-2.105 radians) and Cape Canaveral Air Force Station (-1.412 radians).

Objective functions:

- Launch cost: f_1, millions $.
- Final available mass (excluding primary structure and EPS mass): f_2, kg.
- Theoretical available volume: f_3, m^3.
- Energy consumption related to the thermal control system: f_4, joule.

Constraints:

- Mass constraint: c_1.
- Minimum temperature constraint: c_2.
- Hoop tensile stress constraint: c_3.
- Buckling limit stress constraint: c_4.

This information are then used to define the survey main objectives. In particular the data contained within the modeling framework have been used to set-up the whole analysis. The *Design Variable* class introduced within the modeling infrastructure allows in fact to define the ranges within which the associated variable must be contained as well as the related nominal value (the value that is considered as the current chosen for the baseline and that is used as the initial design point for the survey). These data are then used to create the proper input file that manages the execution of Dakota multidisciplinary platform. In this case an optimization study has been considered and the algorithm parameters for the solving strategy can be selected from the same analysis framework. Such information are then used to consistently generate the input file for the overall optimization cycle. The results obtained from such analysis are reported in the following section and can be directly accessible from the same infrastructure once the whole analysis ends its runs. It is important to underline that the contents of the results themselves are not the primary objective of the current work. Once it has been established that the results are reliable, verifying the correctness of the contained information to avoid wrong links in the data exchange process, the attention is mainly addressed towards the connection with the modeling environment. The main aim of the current work focuses in fact on the feasibility of the integration between a model-based infrastructure and analysis frameworks.

8.4 Results

The previous explicit formulation is then used by the solving platform to lead the exploration of the design space with the final objective to identify the optimal solutions. As introduced in the previous section the optimization process is based on MOGA algorithm and the summary results are provided in the following figures and reports of the iterations cycle. The five reported subcases come from the solution of the same proposed problem but with different settings for the solving method. In particular some of the parameters required to set up the overall process have been changed from one subcase to the other one. The values related to such quantities can be defined within the same environment once the data flow has been provided. Such information is represented for example by the maximum number of iterations, the population size, the mutation type or crossover rate (considering in particular the case of a Genetic Algorithm). All these data are then used to generate the right file required by Dakota to run the whole iterations cycle.
The main purpose of this section is not focused only on the results themselves since they are primarily reported to demonstrate the connection with the model-based environment. The scope is to show how the information gathered within the modeling infrastructure can be used to drive the set up of a multidisciplinary analysis, addressing the efforts to investigate the feasibility of the proposed approach.

8.4.1 Subcase 1

Optimization parameters

The optimization parameters used for the current subcase are reported in table 8.1

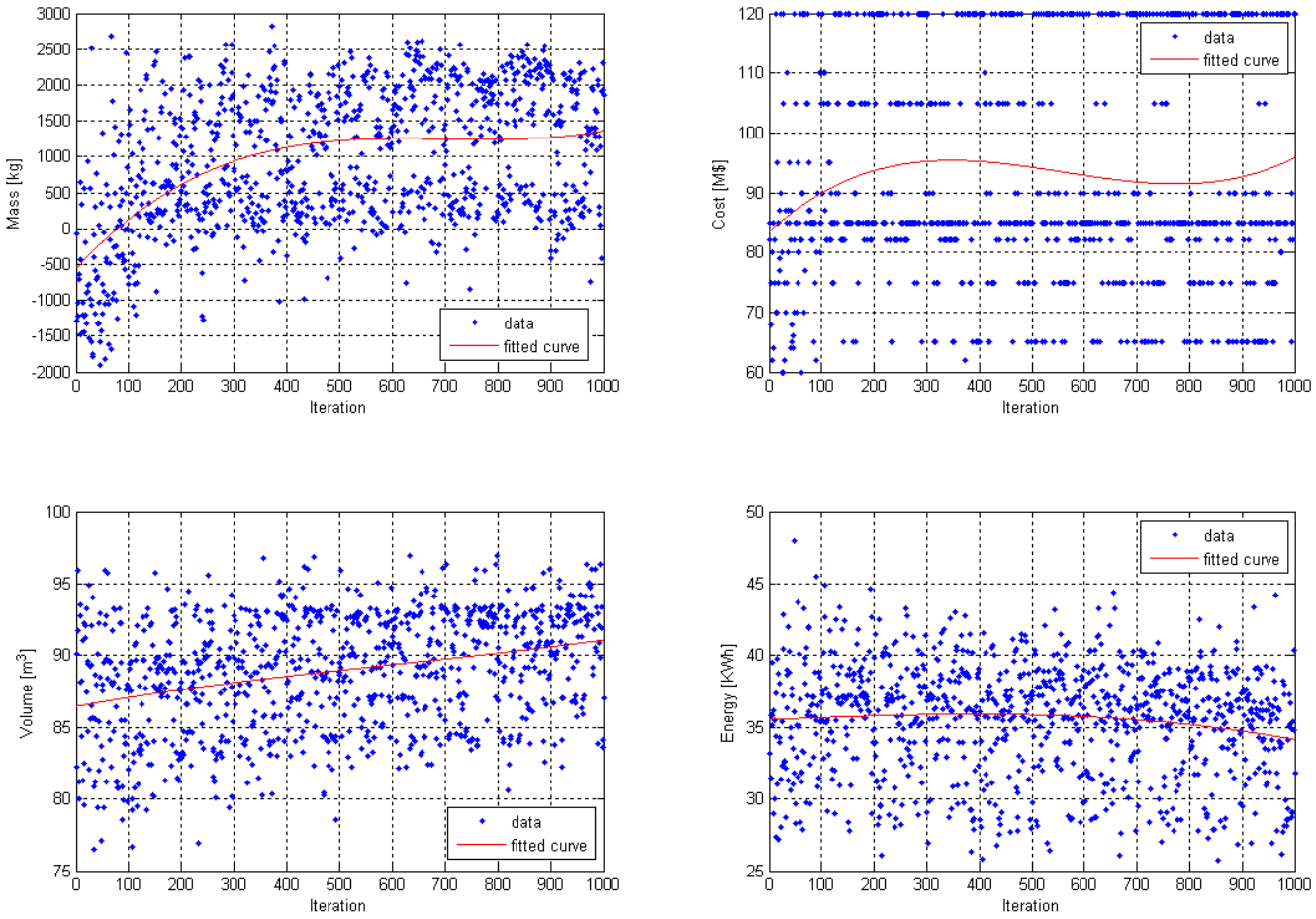

Figure 8.5: Objective functions.

Table 8.1: Parameters of the MOGA method used for the iterations cycle of subcase 1.

Parameter	Value
initialization type	unique random
crossover type	shuffle random
crossover rate	0.8
mutation type	replace uniform
mutation rate	0.2
fitness type	domination count
replacement type	below limit
shrinkage percentage	0.9
percent change (convergence)	0.05

Iterations history

The results related to the objective functions and constraints are reported in the figures 8.5 and 8.6. In particular the quantities are reported with respect to the iteration number and a fitting curve is also introduced to give an approximation of the overall evolution during the iterations cycle.

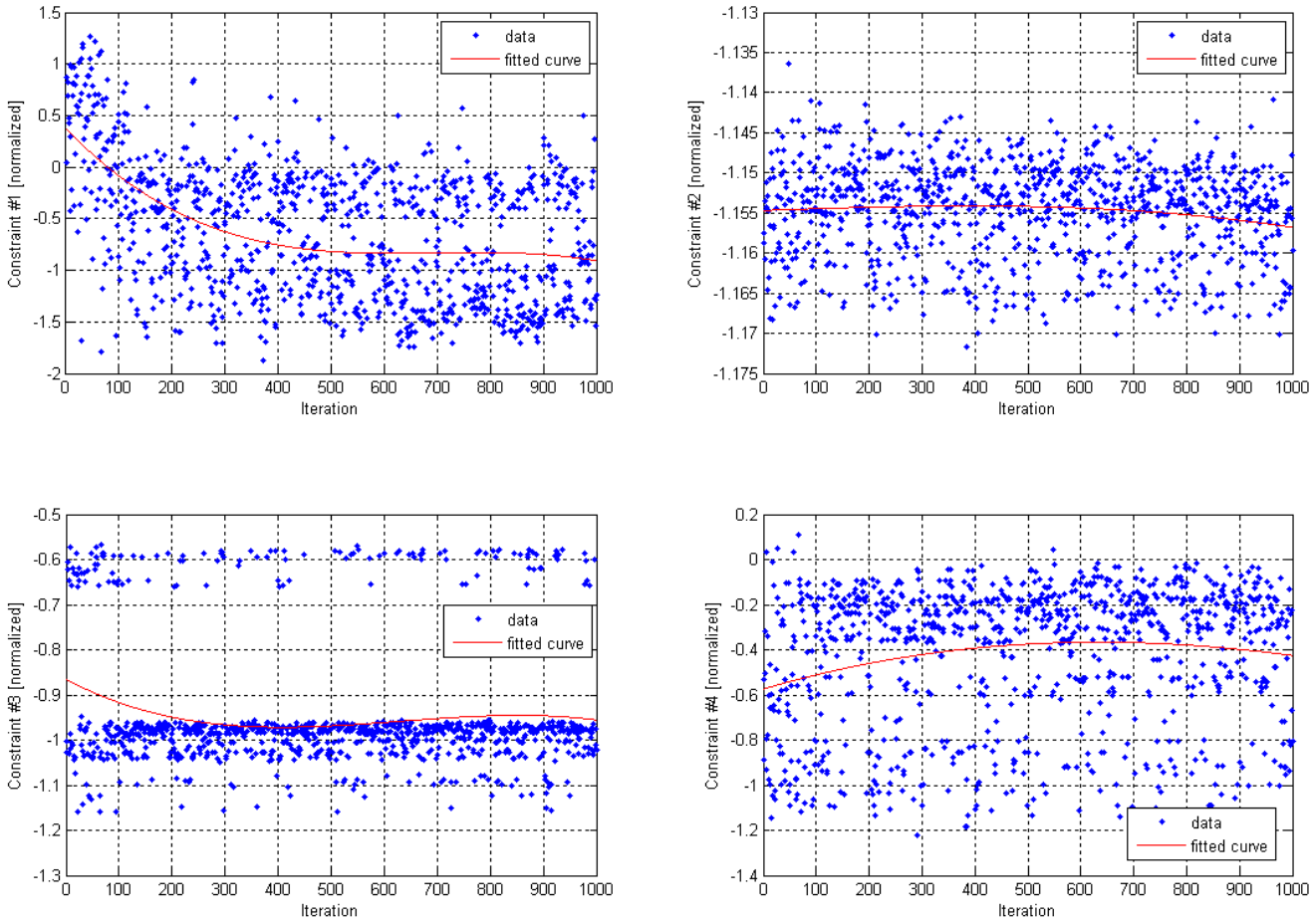

Figure 8.6: Constraints.

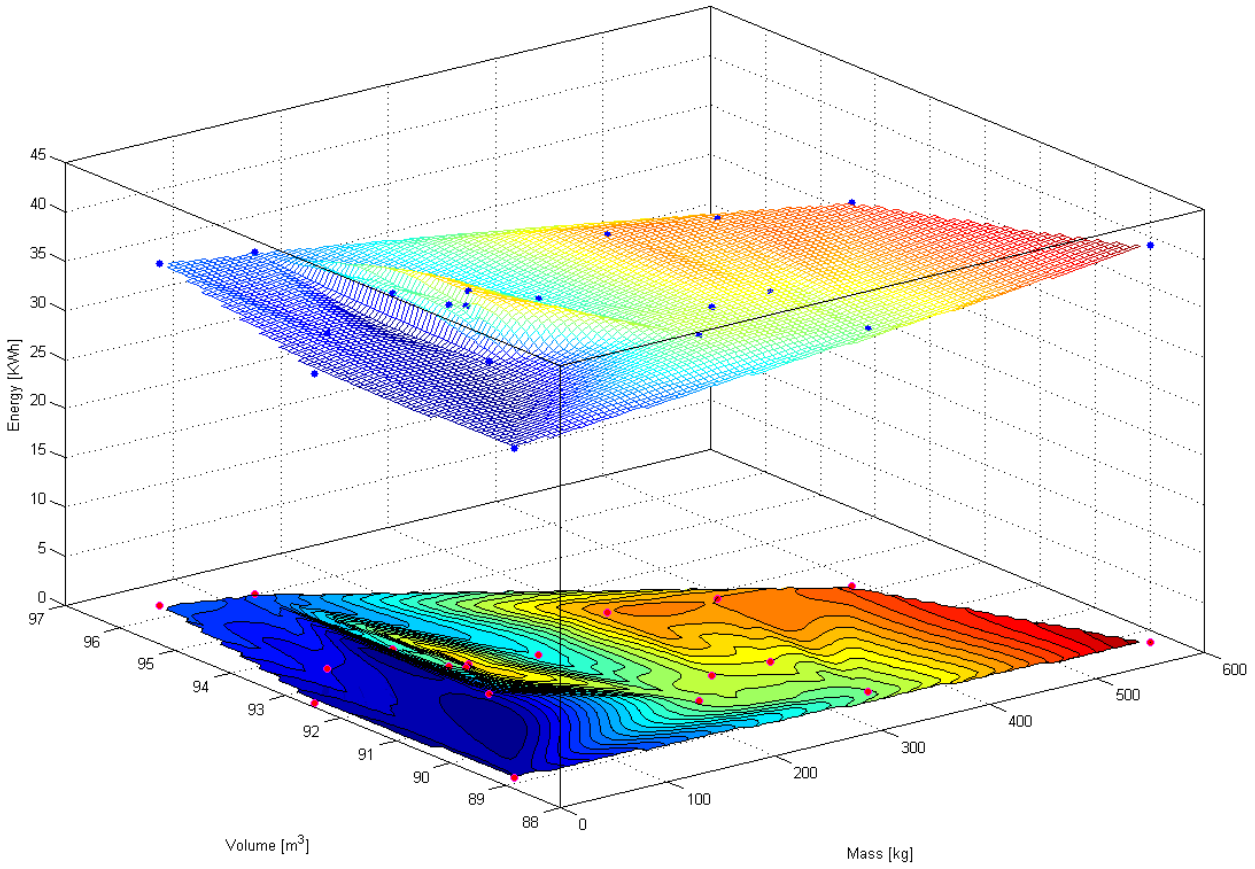

Figure 8.7: Pareto front corresponding to 65 M$ launch cost.

Pareto fronts

The Pareto fronts related to the non-dominated design points are also reported in figures 8.7, 8.8,8.9, 8.10 and 8.11. In particular since four objective functions are available it is difficult to plot them in a readable way. For this reason three of them are reported in the 3d plots (mass, volume and energy) with respect to a specific launch cost. The launch costs corresponding to the non-dominated solutions identified by the algorithm can in fact be used as a parameter in such representation (the discrete range of launch cost helps to pursue such data representation). In this way each cost has its corresponding Pareto front reported in three dimensions.

Optimal design points summary

Some of the non-dominated solutions identified by the solving algorithm are reported in tables 8.2 and 8.3.

8.4.2 Subcase 2

Optimization parameters

The optimization parameters used for the current subcase are reported in table 8.4

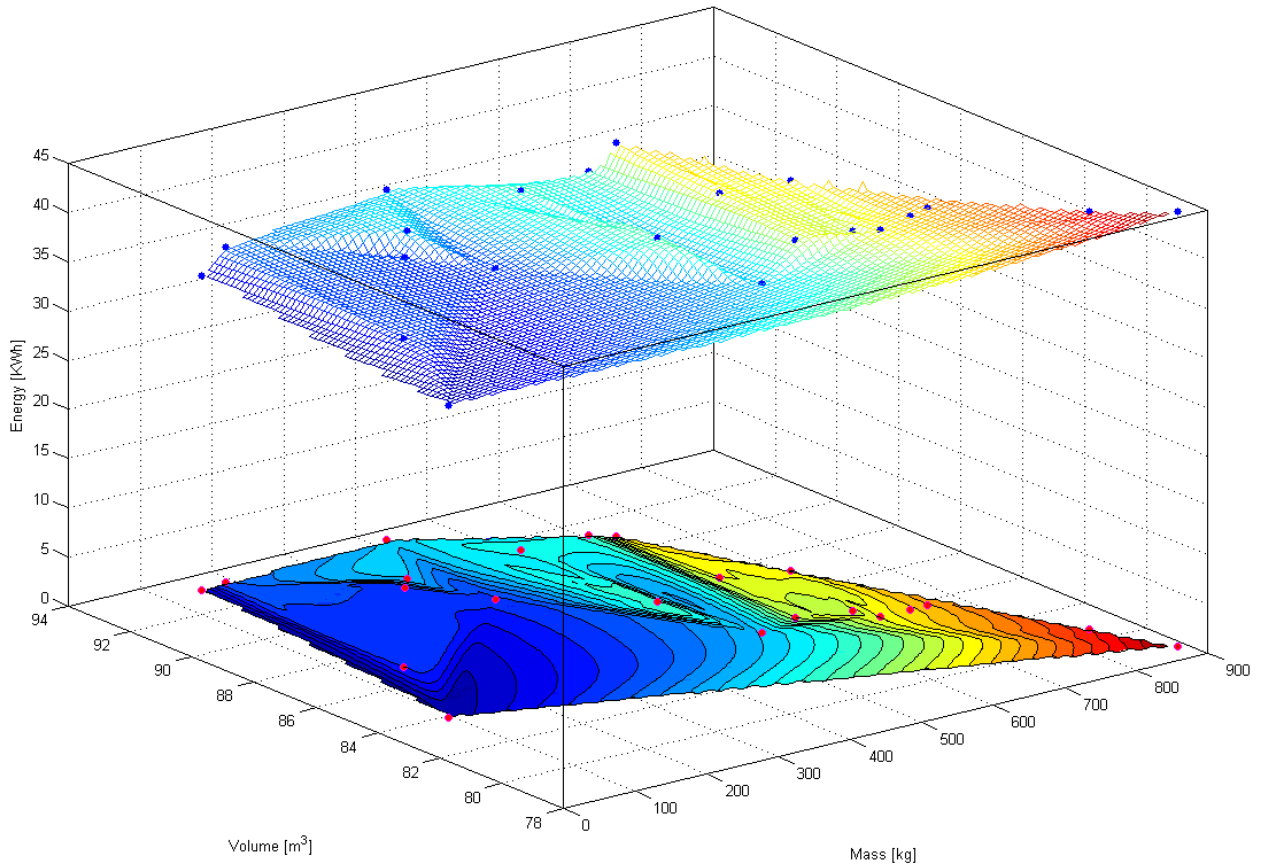

Figure 8.8: Pareto front corresponding to 75 M$ launch cost.

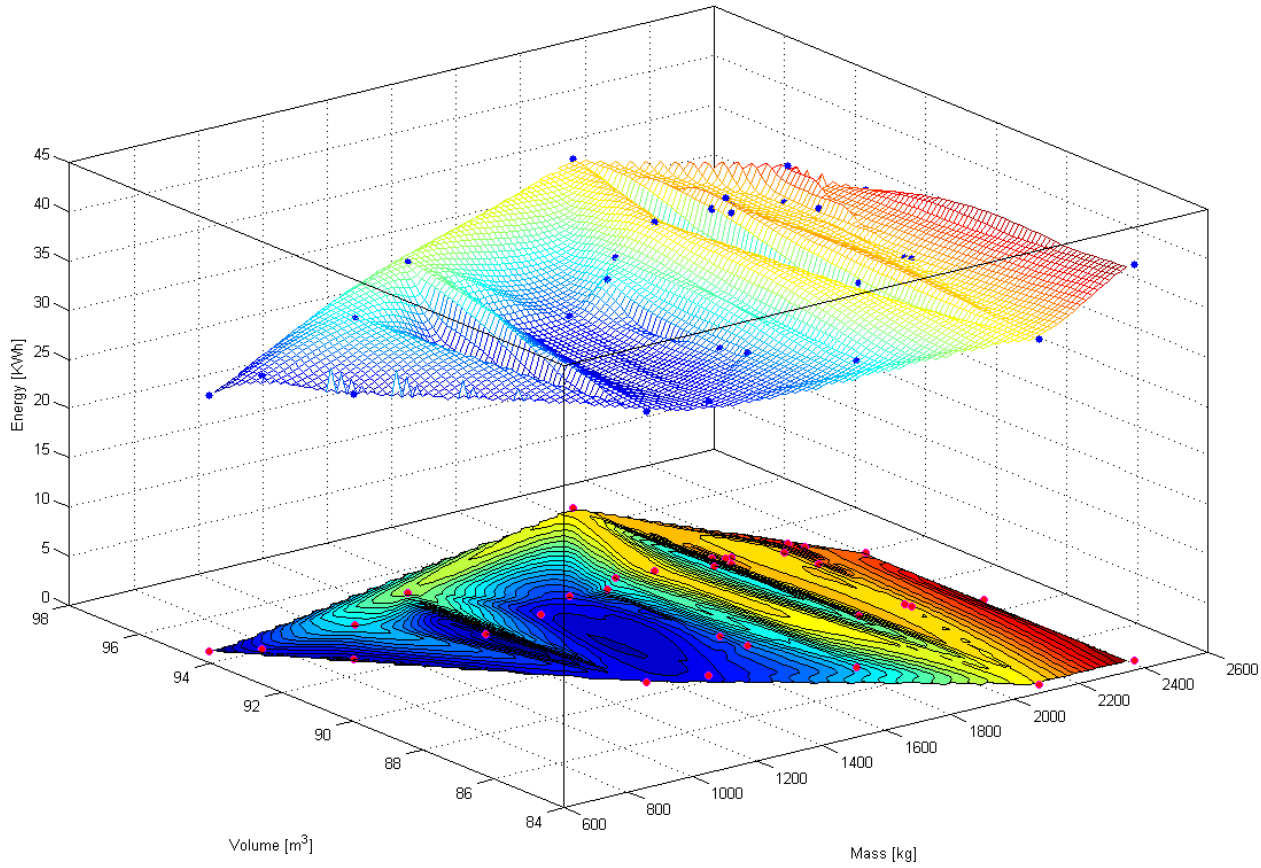

Figure 8.9: Pareto front corresponding to 85 M$ launch cost.

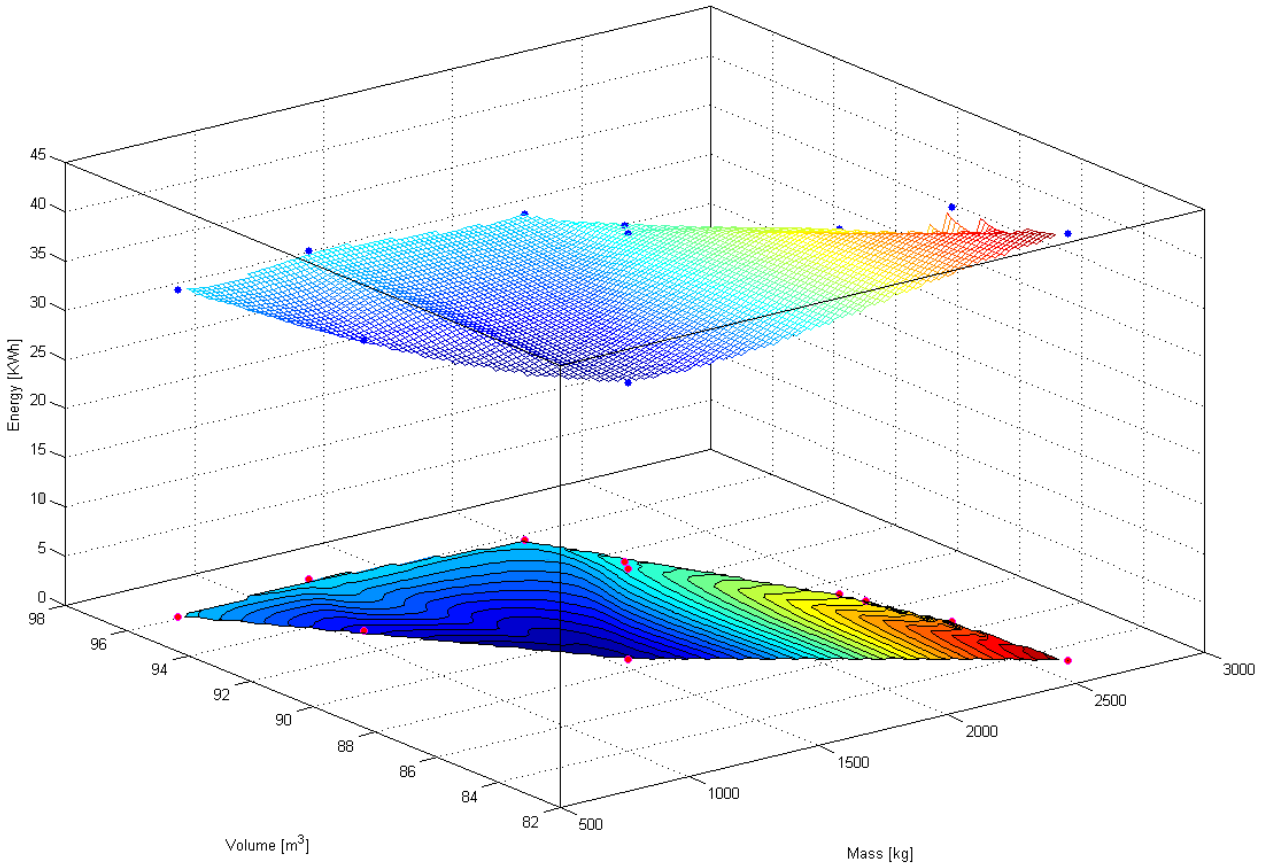

Figure 8.10: Pareto front corresponding to 90 M$ launch cost.

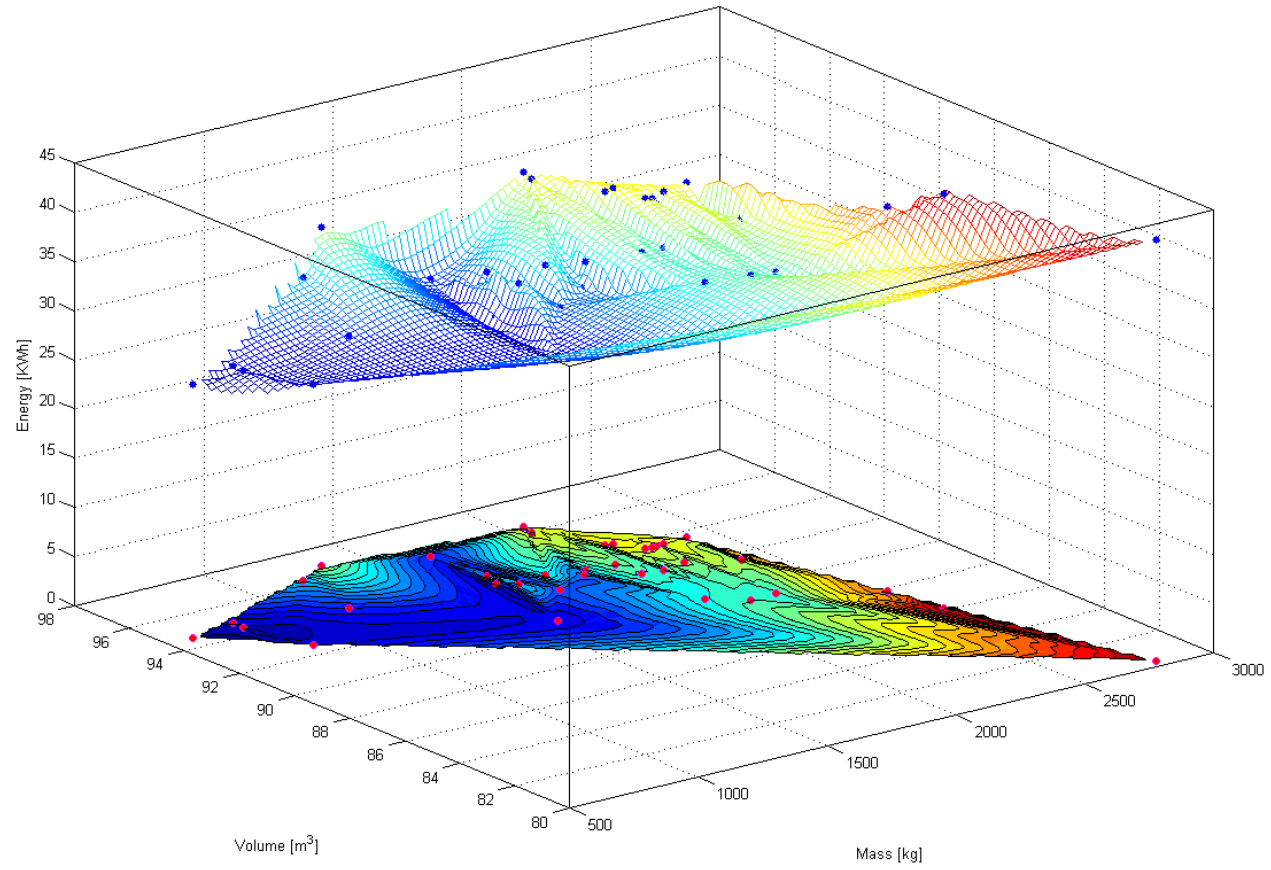

Figure 8.11: Pareto front corresponding to 120 M$ launch cost.

Table 8.2: Some of the non-dominated design points: design variables (subcase 1).

ID.	H (Km)	Sp (sec)	D (m)	L (m)	Thk (m)	P (W)	La	Al	C	B
1	575.15	297.8	4.39	5.9	0.106	1886.3	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 6061 T6	Thin films	NaS
2	575.15	294.3	4.48	5.9	0.106	1886.3	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 6061 T6	Thin films	NaS
3	776.14	297.8	4.47	5.96	0.106	1920.8	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 6061 T6	Thin films	NiH2 CPV
4	647.32	294.3	4.39	5.93	0.11	1580.6	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 2219 T851	Thin films	NaS
5	729.72	294.3	4.48	5.8	0.115	1950.5	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 6061 T6	Thin films	NaS
6	729.72	294.3	4.49	5.94	0.106	1886.3	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 6061 T6	Thin films	NiH2 IPV
7	1013.82	294.3	4.48	5.8	0.11	1915.2	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 2219 T851	Thin films	NaS
8	932.18	294.3	4.39	5.99	0.109	1626.5	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 7075 T73	Thin films	NiMH
9	922.08	297.8	4.47	5.93	0.109	1839.8	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 2219 T851	Thin films	NiMH
10	1294.99	297.8	4.49	5.91	0.109	1920.8	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 6061 T6	GaAs dual junction	NiH2 IPV
81	695.97	294.3	4.39	5.54	0.109	1886.3	Atlas5 V-552 Circular Double Burn	Aluminum 6061 T6	Thin films	NaS
82	695.97	294.3	4.48	5.54	0.109	1886.3	Atlas5 V-552 Circular Double Burn	Aluminum 6061 T6	Thin films	NaS
83	575.15	297.8	4.39	5.9	0.115	1500.7	Atlas5 V-552 Circular Double Burn	Aluminum 2219 T851	Thin films	NaS
84	575.15	297.8	4.39	5.96	0.11	1500.7	Atlas5 V-552 Circular Double Burn	Aluminum 7075 T73	Thin films	NiH2 IPV
85	776.14	294.6	4.49	5.94	0.115	1742.3	Atlas5 V-552 Circular Double Burn	Aluminum 2219 T851	Silicon	NaS
86	729.72	297.8	4.49	5.94	0.115	1654.5	Atlas5 V-552 Circular Double Burn	Aluminum 2219 T851	GaAs dual junction	NiH2 IPV
87	776.14	294.6	4.54	5.99	0.115	1950.5	Atlas5 V-552 Circular Double Burn	Aluminum 2219 T851	Thin films	NiH2 IPV
88	729.72	297.8	4.39	5.93	0.148	1626.5	Atlas5 V-552 Circular Double Burn	Aluminum 7075 T73	GaAs dual junction	NiH2 IPV
89	1013.82	264.2	4.57	5.91	0.115	1654.5	Atlas5 V-552 Circular Double Burn	Aluminum 2219 T851	Silicon	NaS
90	1013.82	275.5	4.49	5.94	0.133	1580.6	Atlas5 V-552 Circular Double Burn	Aluminum 2219 T851	Silicon	NiH2 CPV
91	1013.82	240.3	4.54	5.99	0.115	1950.5	Atlas5 V-552 Circular Double Burn	Aluminum 2219 T851	Silicon	NaS
92	575.15	297.8	4.27	5.65	0.106	1580.6	Ariane5ES LEO i48	Aluminum 2219 T851	Thin films	NaS
93	575.15	297.8	4.47	5.54	0.109	1915.3	Ariane5ES LEO i48	Aluminum 6061 T6	GaAs dual junction	NiH2 CPV
94	776.14	297.8	4.46	5.65	0.109	1839.8	Ariane5ES LEO i48	Aluminum 2219 T851	Thin films	NaS
95	648.05	294.3	4.48	5.99	0.106	1580.6	Ariane5ES LEO i48	Aluminum 2219 T851	Thin films	NiH2 CPV
96	575.15	297.8	4.46	5.9	0.115	1500.7	Ariane5ES LEO i48	Aluminum 2219 T851	Silicon	NaS
97	1013.82	297.8	4.49	5.94	0.109	1915.3	Ariane5ES LEO i48	Aluminum 2219 T851	Silicon	NaS
98	659.07	297.8	4.39	5.94	0.124	1500.7	Ariane5ES LEO i48	Aluminum 2219 T851	Thin films	NaS
99	922.08	294.3	4.49	5.94	0.115	1839.8	Ariane5ES LEO i48	Aluminum 7075 T73	Thin films	NaS
100	575.15	291.0	4.49	5.94	0.11	1580.6	Ariane5ES LEO i60	Aluminum 2219 T851	GaInP dual junction	NaS
101	776.14	294.6	4.44	5.99	0.115	1500.7	Ariane5ES LEO i48	Aluminum 2219 T851	Silicon	NaS

Table 8.3: Some of the non-dominated design points: objective functions and constraints (subcase 1).

ID.	Cost (M$)	Mass (kg)	Volume (m^3)	Energy (KWh)	con. 1	con. 2	con. 3	con. 4
1	65	582.53	88.66	40.32	-0.39	-1.15	-0.59	-0.1
2	65	487.29	92.22	39.07	-0.32	-1.15	-0.58	-0.04
3	65	391.72	92.8	38.52	-0.26	-1.15	-0.58	-0.05
4	65	346.65	89.18	36.9	-0.23	-1.15	-0.98	-0.25
5	65	331.98	90.67	37.6	-0.22	-1.15	-0.6	-0.25
6	65	306.52	93.13	38.45	-0.2	-1.15	-0.58	-0.04
7	65	278.59	90.7	37.34	-0.19	-1.15	-0.97	-0.2
8	65	235.25	90.08	37.03	-0.16	-1.15	-1.09	-0.22
9	65	206.76	92.44	36.09	-0.14	-1.15	-0.97	-0.19
10	65	152.16	92.64	37.88	-0.1	-1.15	-0.59	-0.12
81	90	2618.73	83.25	43.32	-1.75	-1.15	-0.59	-0.16
82	90	2568.14	86.6	42.01	-1.71	-1.15	-0.59	-0.11
83	90	2477.91	88.61	35.76	-1.65	-1.15	-0.99	-0.35
84	90	2453.28	89.28	37.12	-1.64	-1.15	-1.09	-0.21
85	90	2122.54	93.46	34.15	-1.42	-1.16	-0.98	-0.29
86	90	2089.94	93.08	34.09	-1.39	-1.16	-0.98	-0.29
87	90	2005.31	95.72	33.2	-1.34	-1.16	-0.98	-0.27
88	90	1551.47	88.59	28.09	-1.03	-1.17	-1.16	-1.06
89	90	1248.5	96.38	33.37	-0.83	-1.16	-0.98	-0.24
90	90	1073.89	93.15	29.51	-0.72	-1.16	-1.02	-0.71
91	90	705.4	96.1	33.24	-0.47	-1.16	-0.98	-0.27
92	120	2814.76	80.35	42.78	-1.88	-1.14	-0.97	-0.22
93	120	2604.97	86.14	42.09	-1.74	-1.15	-0.59	-0.11
94	120	2568.0	87.84	39.09	-1.71	-1.15	-0.97	-0.18
95	120	2398.39	93.57	36.13	-1.6	-1.15	-0.96	-0.09
96	120	2397.26	91.56	34.77	-1.6	-1.15	-0.98	-0.31
97	120	2303.71	93.49	35.79	-1.54	-1.15	-0.97	-0.17
98	120	2282.47	89.21	32.72	-1.52	-1.16	-1.01	-0.58
99	120	2273.62	93.46	34.83	-1.52	-1.16	-1.1	-0.27
100	120	2256.45	93.49	35.42	-1.5	-1.15	-0.97	-0.17
101	120	2245.34	92.21	34.12	-1.5	-1.16	-0.98	-0.32

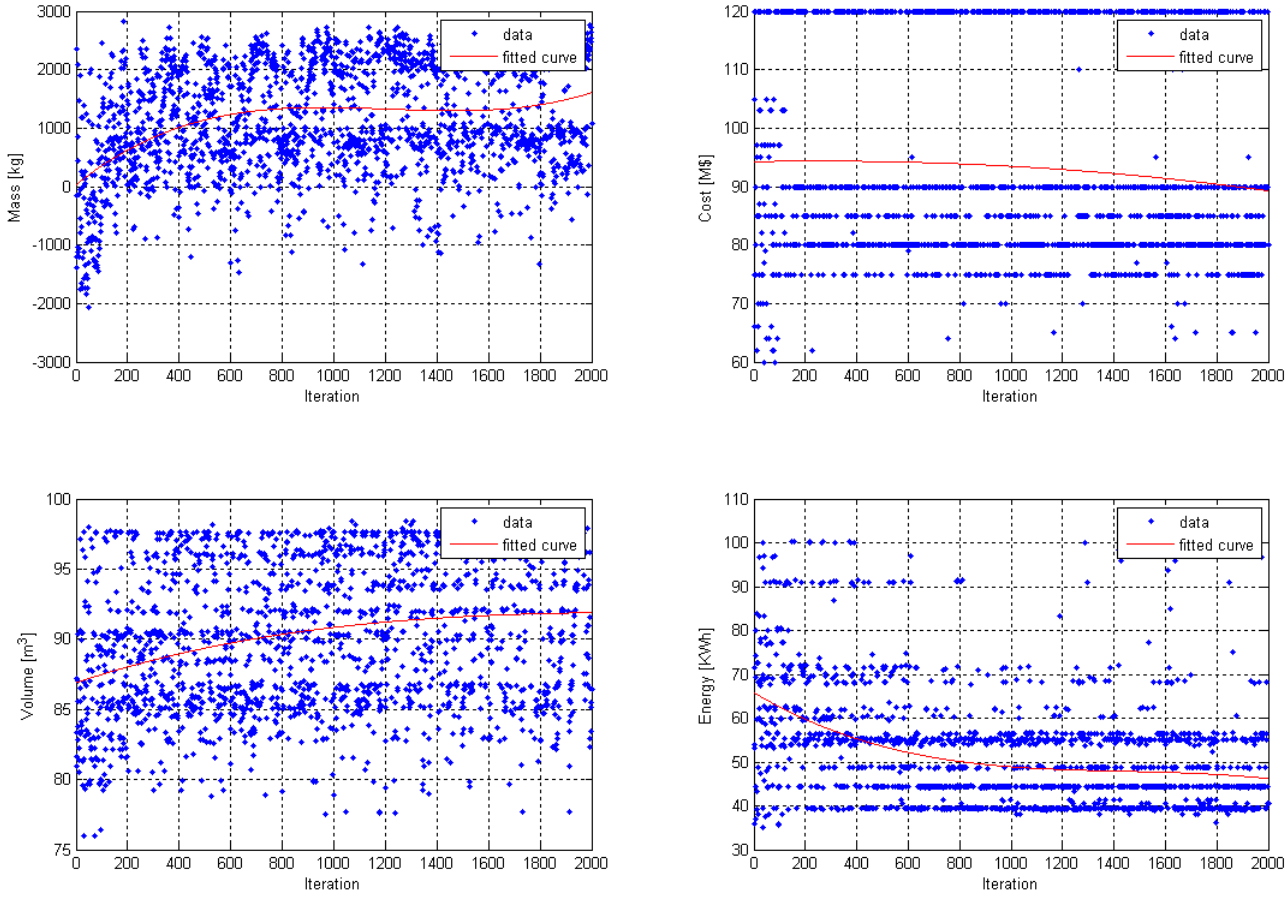

Figure 8.12: Objective functions.

Table 8.4: Parameters of the MOGA method used for the iterations cycle of subcase 2.

Parameter	Value
initialization type	simple random
crossover type	multi point binary
crossover rate	0.7
mutation type	bit random
mutation rate	0.3
fitness type	layer rank
replacement type	elitist
shrinkage percentage	0.9
percent change (convergence)	0.05

Iterations history

The results related to the objective functions and constraints are reported in the figures 8.12 and 8.13. In particular the quantities are reported with respect to the iteration number and a fitting curve is also introduced to give an approximation of the overall evolution during the iterations cycle.

Pareto fronts

The Pareto fronts related to the non-dominated design points are also reported in figures 8.14, 8.15,8.16 and 8.17. In particular since four objective functions are available it is difficult to plot them in a readable way. For this reason three of them are reported in the 3d plots (mass, volume and energy) with respect

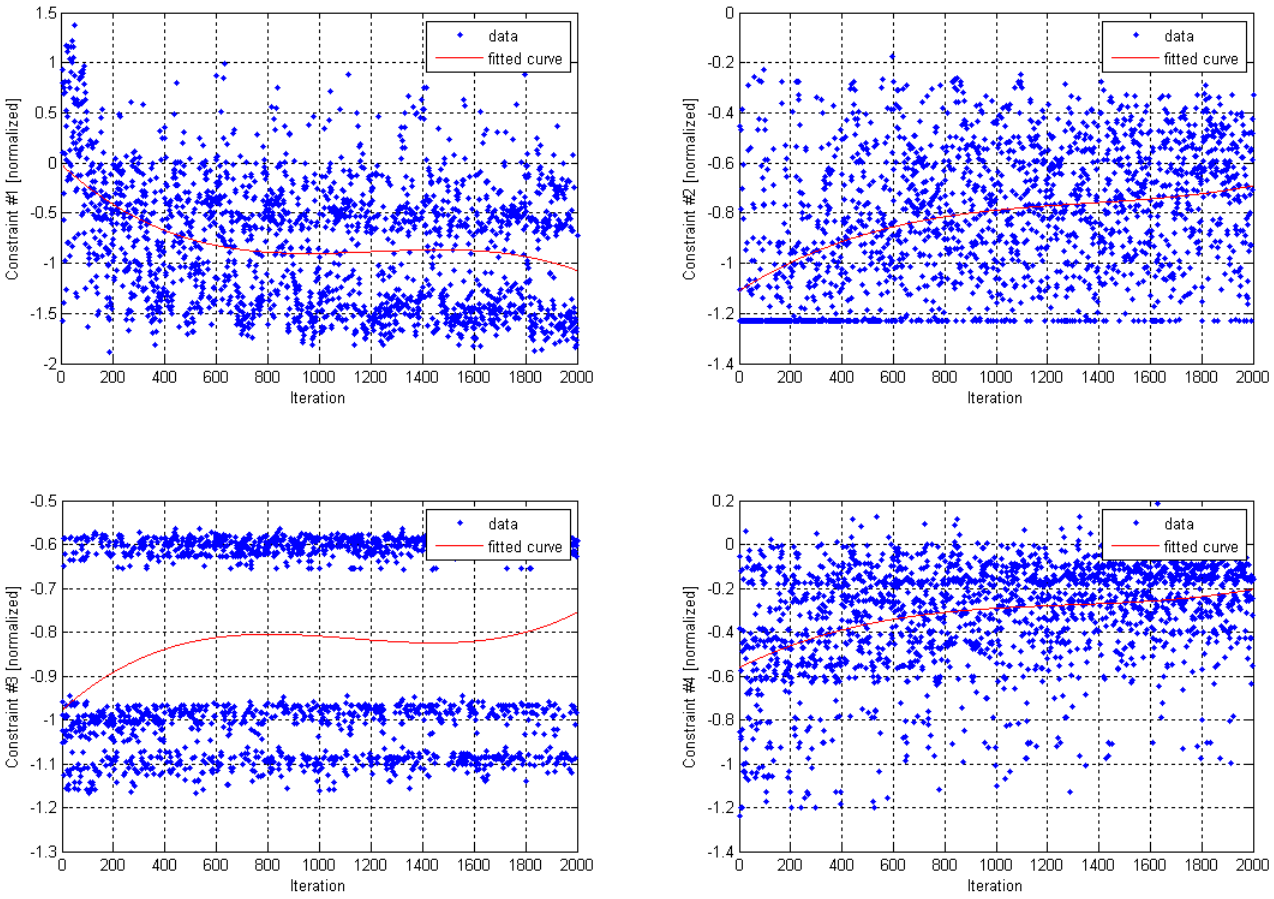

Figure 8.13: Constraints.

to a specific launch cost. The launch costs corresponding to the non-dominated solutions identified by the algorithm can in fact be used as a parameter in such representation (the discrete range of launch cost helps to pursue such data representation). In this way each cost has its corresponding Pareto front reported in three dimensions.

Optimal design points summary

Some of the non-dominated solutions identified by the solving algorithm are reported in tables 8.5 and 8.6.

8.4.3 Subcase 3

Optimization parameters

The optimization parameters used for the current subcase are reported in table 8.7

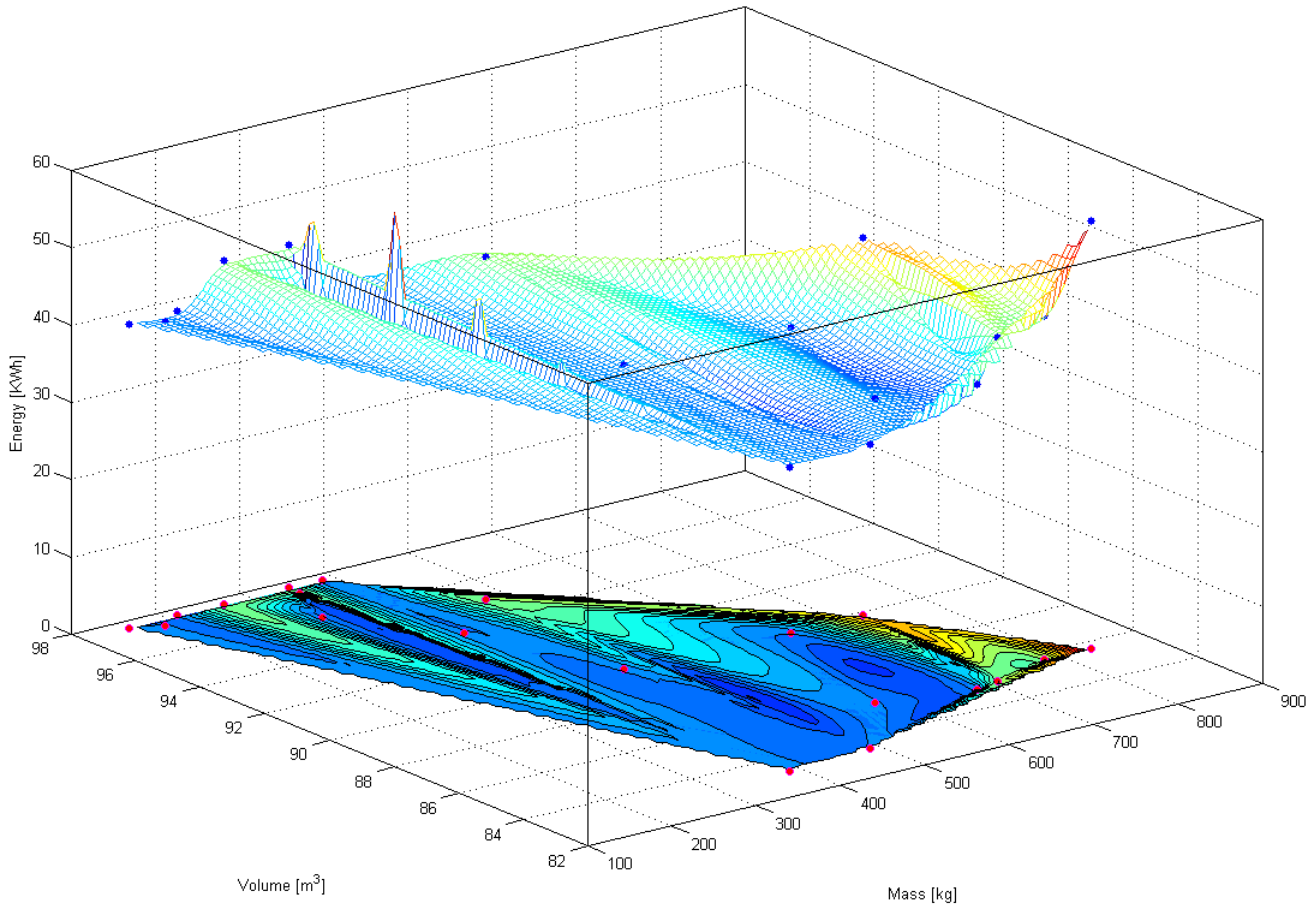

Figure 8.14: Pareto front corresponding to 75 M$ launch cost.

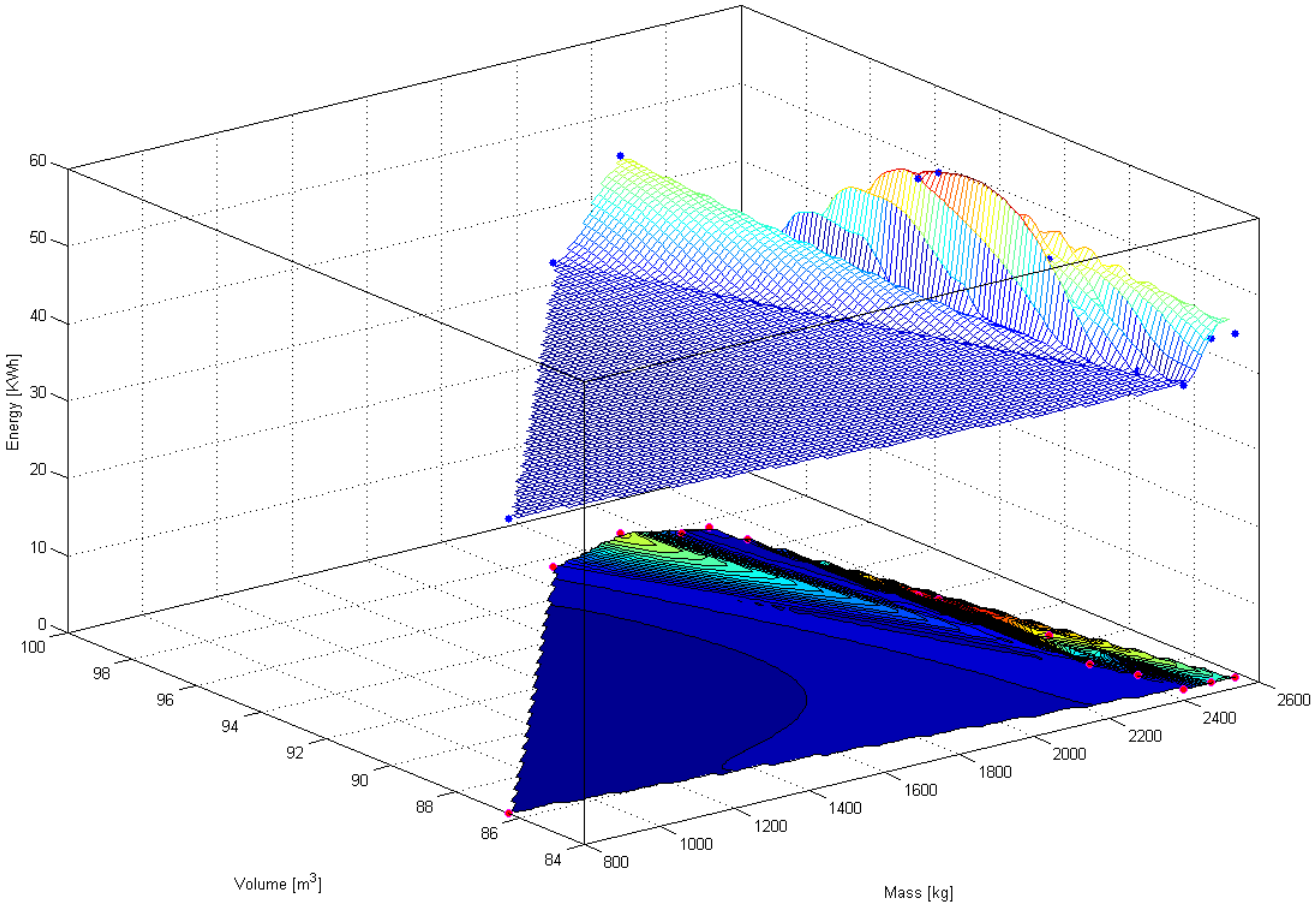

Figure 8.15: Pareto front corresponding to 85 M$ launch cost.

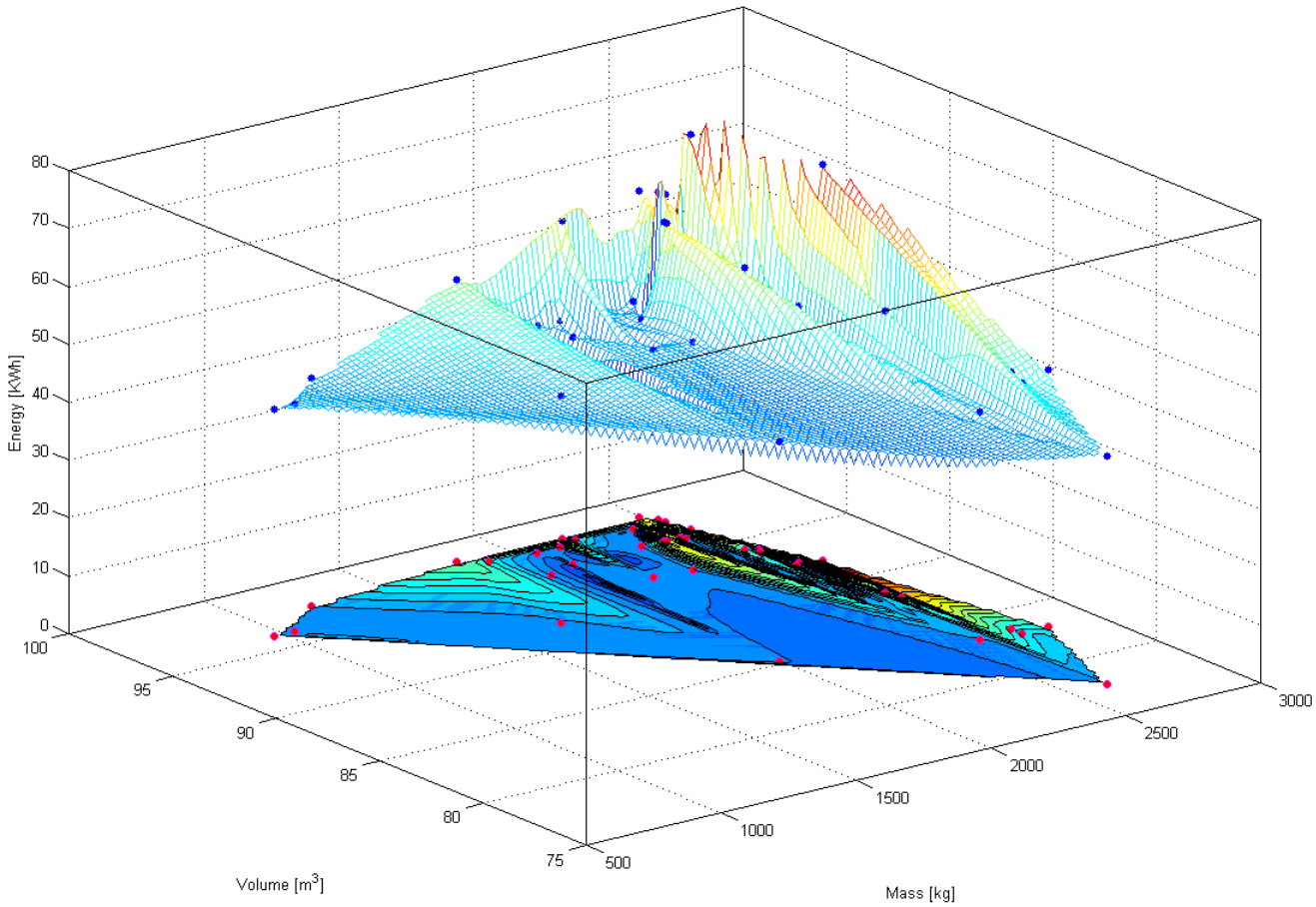

Figure 8.16: Pareto front corresponding to 90 M$ launch cost.

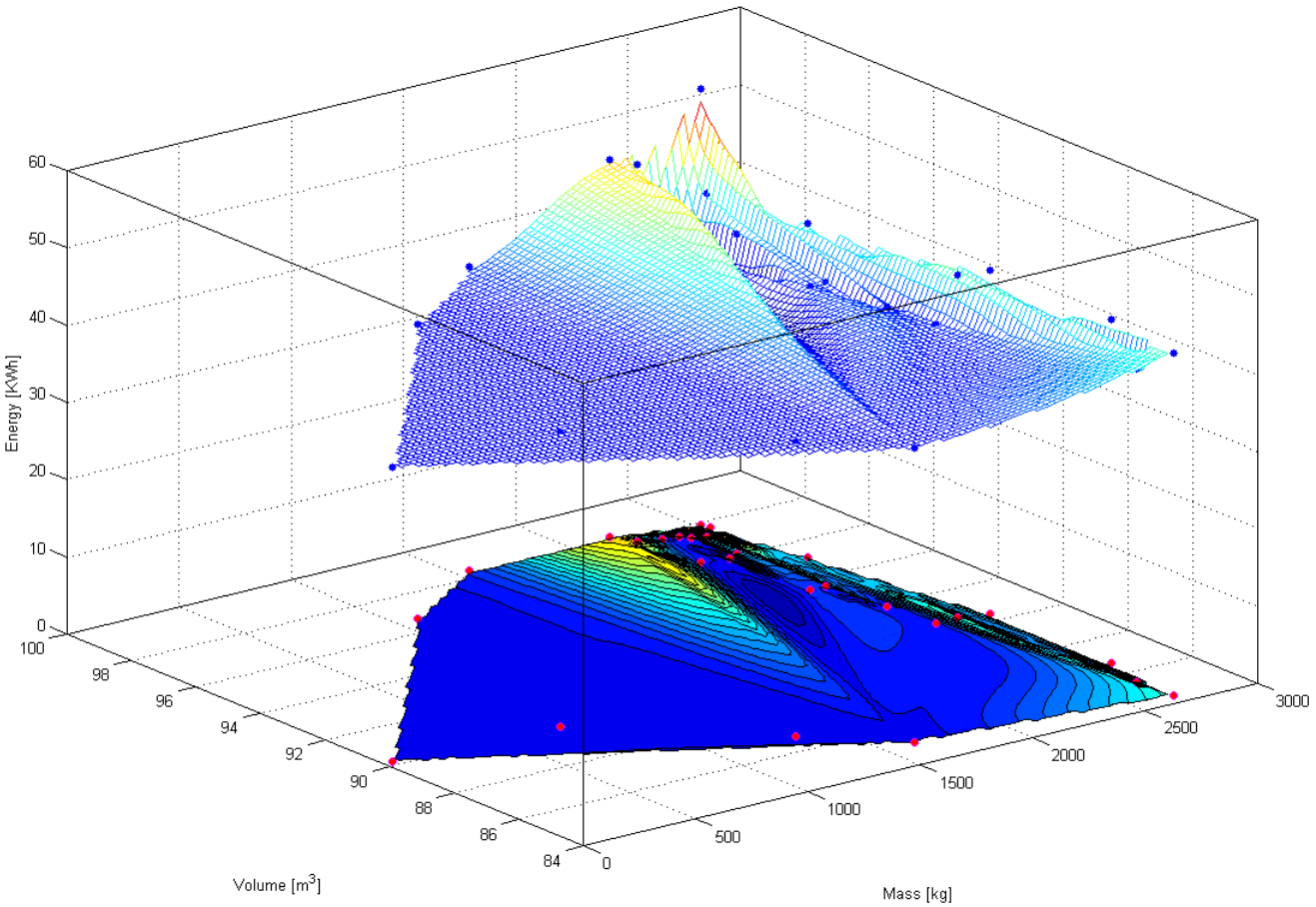

Figure 8.17: Pareto front corresponding to 120 M$ launch cost.

Table 8.5: Some of the non-dominated design points: design variables (subcase 2).

ID.	H (Km)	Sp (sec)	D (m)	L (m)	Thk (m)	P (W)	La	Al	C	B
1	453.77	290.2	4.35	5.88	0.103	849.4	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 7075 T73	GaInP dual junction	NaS
2	453.77	290.2	4.58	5.99	0.111	849.4	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 7075 T73	Thin films	NaS
3	453.77	286.2	4.58	5.76	0.111	575.9	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 6061 T6	Silicon	NiMH
4	648.91	296.2	4.58	5.95	0.111	874.2	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 7075 T73	Thin films	NaS
5	555.55	296.2	4.58	5.95	0.111	874.2	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 7075 T73	Silicon	NaS
6	453.77	296.2	4.29	5.97	0.103	849.4	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 6061 T6	Silicon	NaS
7	453.77	299.2	4.29	5.97	0.103	659.3	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 2219 T851	Silicon	NaS
8	453.77	296.2	4.45	5.83	0.103	735.1	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 7075 T73	GaAs dual junction	NaS
9	434.36	290.2	4.31	5.87	0.103	659.3	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 7075 T73	GaAs dual junction	NaS
10	1059.93	296.2	4.31	5.87	0.103	575.9	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 2219 T851	GaAs dual junction	NaS
72	453.77	299.2	4.35	5.74	0.111	659.3	Atlas5 V-552 Circular Single Burn	Aluminum 6061 T6	Thin films	NaS
73	1006.16	299.2	4.35	5.74	0.107	659.3	Atlas5 V-552 Circular Single Burn	Aluminum 6061 T6	GaInP dual junction	NaS
74	453.77	299.2	4.35	5.74	0.111	575.9	Atlas5 V-552 Circular Single Burn	Aluminum 7075 T73	Silicon	NaS
75	453.77	299.2	4.45	5.76	0.112	735.1	Atlas5 V-552 Circular Single Burn	Aluminum 6061 T6	Silicon	NaS
76	434.36	299.2	4.31	5.91	0.112	575.9	Atlas5 V-552 Circular Single Burn	Aluminum 2219 T851	GaInP dual junction	Li Ion
77	453.77	299.2	4.45	5.95	0.107	849.4	Atlas5 V-552 Circular Single Burn	Aluminum 7075 T73	Silicon	NaS
78	648.91	296.2	4.32	5.97	0.111	575.9	Atlas5 V-552 Circular Single Burn	Aluminum 6061 T6	GaAs dual junction	NaS
79	495.71	296.2	4.58	5.63	0.112	831.7	Atlas5 V-552 Circular Single Burn	Aluminum 6061 T6	Thin films	NaS
80	434.36	299.2	4.58	5.91	0.112	575.9	Atlas5 V-552 Circular Single Burn	Aluminum 6061 T6	GaAs dual junction	NaS
81	453.77	299.2	4.58	5.97	0.111	575.9	Atlas5 V-552 Circular Single Burn	Aluminum 6061 T6	Silicon	NaS
125	434.36	299.2	4.31	5.97	0.107	659.3	Ariane5ES LEO i60	Aluminum 6061 T6	GaAs dual junction	NaS
126	453.77	299.2	4.41	5.97	0.107	659.3	Ariane5ES LEO i60	Aluminum 6061 T6	Thin films	NaS
127	453.77	299.2	4.31	5.84	0.111	659.3	Ariane5ES LEO i60	Aluminum 6061 T6	GaAs dual junction	Li Ion
128	495.71	296.2	4.32	5.88	0.111	594.8	Ariane5ES LEO i48	Aluminum 6061 T6	GaInP dual junction	Li Ion
129	495.71	296.2	4.45	5.88	0.112	659.3	Ariane5ES LEO i48	Aluminum 6061 T6	Thin films	Li Ion
130	453.77	299.2	4.45	5.97	0.107	659.3	Ariane5ES LEO i48	Aluminum 7075 T73	Silicon	NaS
131	453.77	299.2	4.58	5.8	0.111	575.9	Ariane5ES LEO i48	Aluminum 6061 T6	GaAs dual junction	NaS
132	453.77	299.2	4.58	5.97	0.107	659.3	Ariane5ES LEO i48	Aluminum 7075 T73	Silicon	NaS
133	464.67	299.2	4.58	5.99	0.111	874.2	Ariane5ES LEO i48	Aluminum 6061 T6	GaAs dual junction	Li Ion
134	434.36	299.2	4.45	5.88	0.119	567.3	Ariane5ES LEO i60	Aluminum 6061 T6	Thin films	NaS

Table 8.6: Some of the non-dominated design points: objective functions and constraints (subcase 2).

ID.	Cost (M$)	Mass (kg)	Volume (m^3)	Energy (KWh)	con. 1	con. 2	con. 3	con. 4
1	65	482.54	86.62	55.26	-0.32	-0.81	-1.08	-0.07
2	65	165.32	97.89	55.12	-0.11	-0.94	-1.09	-0.15
3	65	54.26	94.04	39.34	-0.04	-0.59	-0.59	-0.11
4	70	385.06	97.29	56.46	-0.26	-1.0	-1.09	-0.15
5	70	324.74	97.29	56.45	-0.22	-1.0	-1.09	-0.15
6	75	836.58	85.63	55.35	-0.56	-0.69	-0.58	-0.06
7	75	780.52	85.63	44.48	-0.52	-0.4	-0.96	-0.12
8	75	725.33	89.81	48.84	-0.48	-0.68	-1.07	-0.0
9	75	707.51	85.16	44.46	-0.47	-0.46	-1.08	-0.08
10	75	675.55	84.99	39.49	-0.45	-0.33	-0.96	-0.11
72	85	2581.84	84.51	44.38	-1.72	-0.65	-0.6	-0.23
73	85	2518.54	84.53	44.48	-1.68	-0.6	-0.59	-0.14
74	85	2443.69	84.51	39.35	-1.63	-0.53	-1.1	-0.27
75	85	2438.71	88.61	48.75	-1.63	-0.82	-0.6	-0.18
76	85	2424.97	85.69	39.37	-1.62	-0.43	-0.98	-0.33
77	85	2402.57	91.62	55.2	-1.6	-0.87	-1.08	-0.11
78	85	2388.91	86.78	39.44	-1.59	-0.33	-0.6	-0.25
79	85	2383.96	92.02	53.98	-1.59	-1.13	-0.59	-0.1
80	85	2328.96	96.65	39.37	-1.55	-0.49	-0.59	-0.1
81	85	2308.28	97.63	39.39	-1.54	-0.45	-0.59	-0.09
125	120	2727.89	86.63	44.48	-1.82	-0.43	-0.59	-0.15
126	120	2723.74	90.35	44.46	-1.82	-0.47	-0.59	-0.1
127	120	2681.87	84.36	44.42	-1.79	-0.56	-0.6	-0.26
128	120	2673.44	85.46	40.56	-1.78	-0.42	-0.6	-0.25
129	120	2611.19	90.57	44.41	-1.74	-0.6	-0.6	-0.18
130	120	2608.79	92.16	44.43	-1.74	-0.52	-1.08	-0.11
131	120	2576.05	94.87	39.35	-1.72	-0.56	-0.59	-0.09
132	120	2528.28	97.66	44.41	-1.69	-0.58	-1.08	-0.04
133	120	2523.62	97.89	56.5	-1.68	-0.95	-0.59	-0.09
134	120	2508.42	90.53	38.81	-1.67	-0.53	-0.61	-0.37

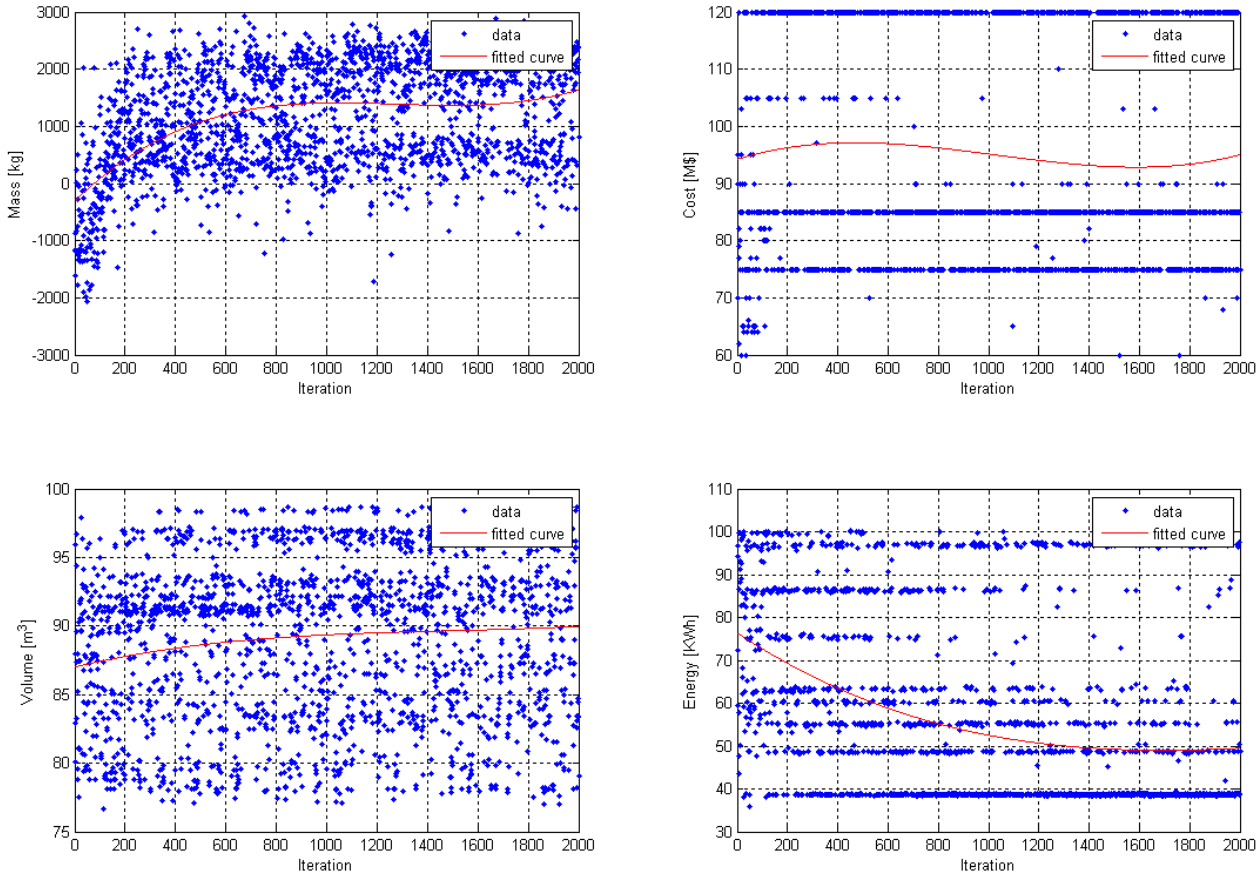

Figure 8.18: Objective functions.

Table 8.7: Parameters of the MOGA method used for the iterations cycle of subcase 3.

Parameter	Value
initialization type	unique random
crossover type	multi point real
crossover rate	0.85
mutation type	offset normal
mutation rate	0.35
fitness type	domination count
replacement type	roulette wheel
shrinkage percentage	0.9
percent change (convergence)	0.04

Iterations history

The results related to the objective functions and constraints are reported in the figures 8.18 and 8.19. In particular the quantities are reported with respect to the iteration number and a fitting curve is also introduced to give an approximation of the overall evolution during the iterations cycle.

Pareto fronts

The Pareto fronts related to the non-dominated design points are also reported in figures 8.20, 8.21,8.22 and 8.23. In particular since four objective functions are available it is difficult to plot them in a readable way. For this reason three of them are reported in the 3d plots (mass, volume and energy) with respect to a specific launch cost. The launch costs corresponding to the non-dominated solutions identified by the

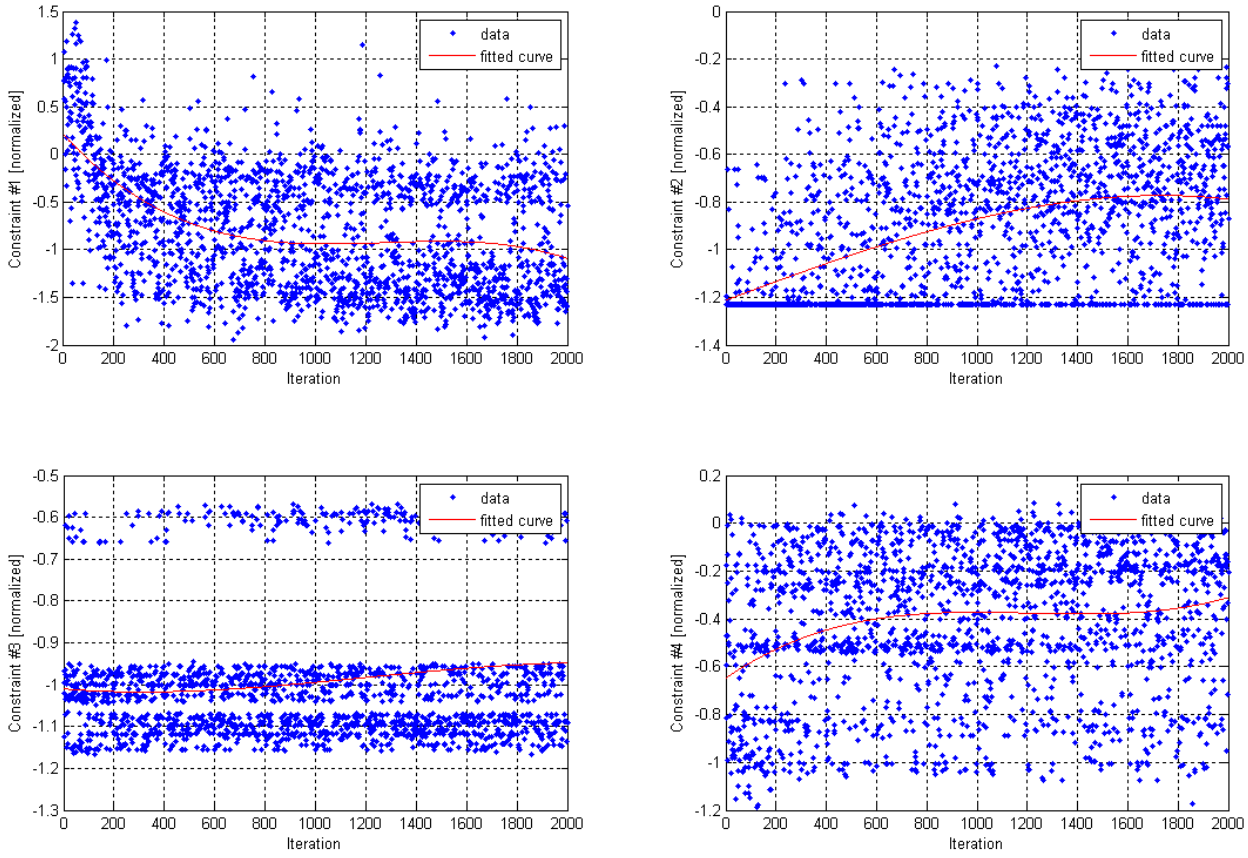

Figure 8.19: Constraints.

algorithm can in fact be used as a parameter in such representation (the discrete range of launch cost helps to pursue such data representation). In this way each cost has its corresponding Pareto front reported in three dimensions.

Optimal design points summary

Some of the non-dominated solutions identified by the solving algorithm are reported in tables 8.8 and 8.9.

8.4.4 Subcase 4

Optimization parameters

The optimization parameters used for the current subcase are reported in table 8.10

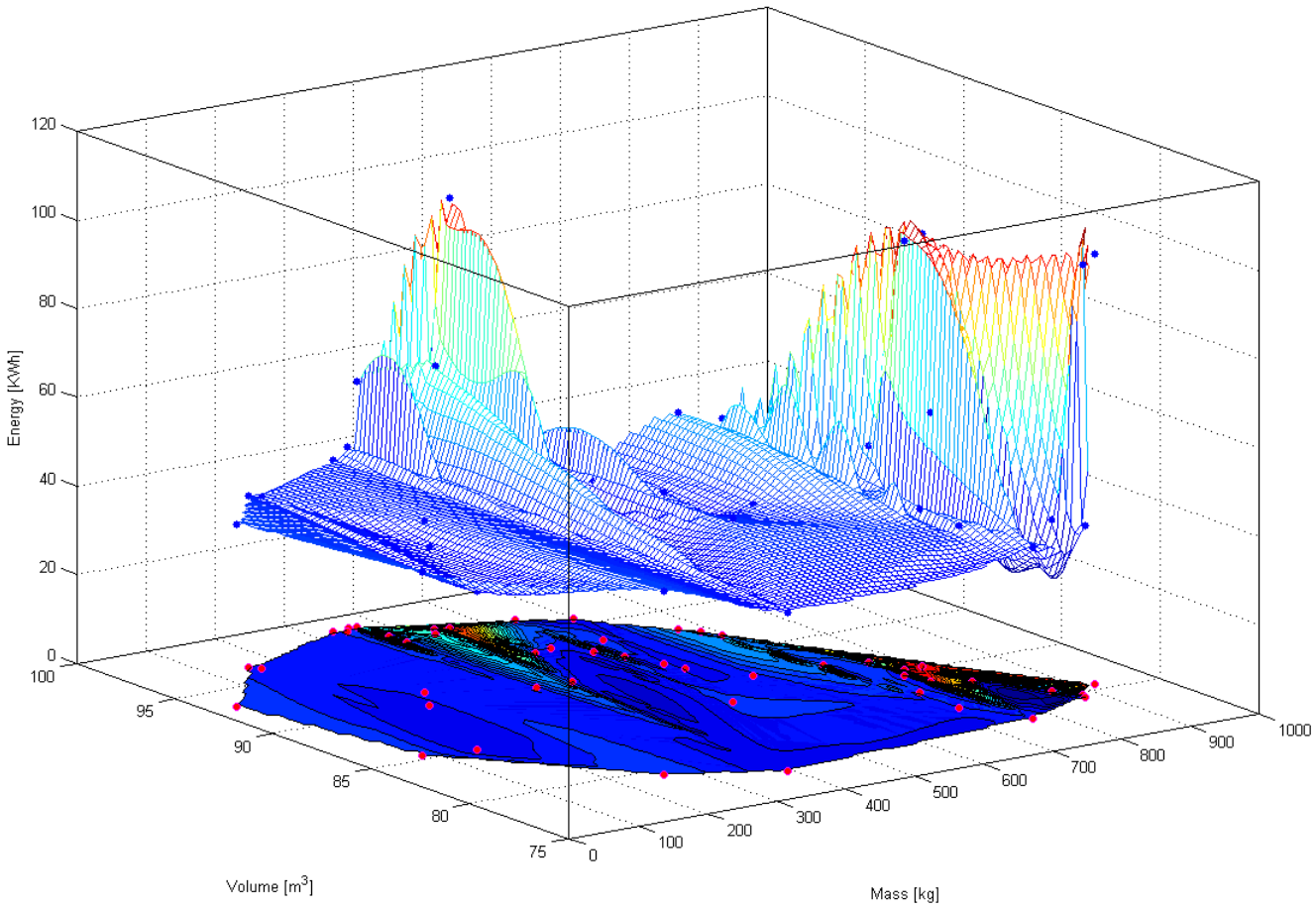

Figure 8.20: Pareto front corresponding to 75 M$ launch cost.

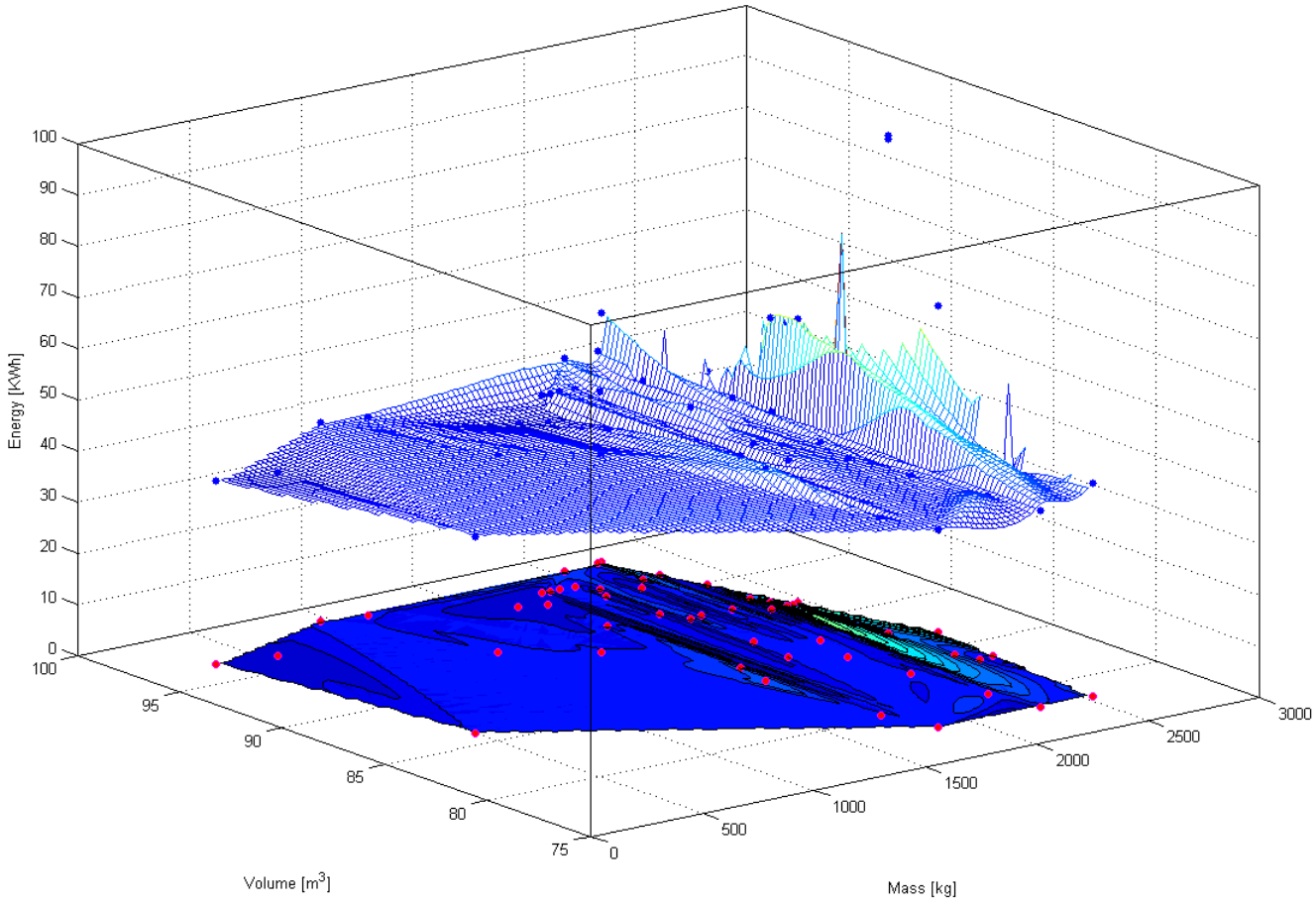

Figure 8.21: Pareto front corresponding to 85 M$ launch cost.

Table 8.8: Some of the non-dominated design points: design variables (subcase 3).

ID.	H (Km)	Sp (sec)	D (m)	L (m)	Thk (m)	P (W)	La	Al	C	B
1	913.2	298.4	4.31	5.57	0.105	1854.2	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 6061 T6	Thin films	NaS
2	1016.35	295.6	4.31	5.52	0.105	1854.2	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 6061 T6	Thin films	Li Ion
3	1119.02	298.4	4.29	5.54	0.106	565.1	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 6061 T6	Thin films	Li Ion
4	1119.02	298.4	4.31	5.57	0.106	565.1	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 6061 T6	Thin films	Li Ion
5	1119.02	298.4	4.31	5.57	0.106	565.1	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 6061 T6	Thin films	Li Ion
6	913.2	295.6	4.26	5.86	0.106	565.1	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 6061 T6	Thin films	Li Ion
7	913.2	295.6	4.31	5.9	0.105	1854.2	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 6061 T6	Thin films	NaS
8	1016.35	298.4	4.45	5.52	0.105	1854.2	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 7075 T73	Thin films	Li Ion
9	627.12	295.6	4.28	5.86	0.105	950.3	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 2219 T851	Thin films	Li Ion
10	1067.29	295.6	4.26	5.54	0.106	565.1	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 2219 T851	Thin films	Li Ion
52	609.42	298.4	4.29	5.54	0.125	565.1	Delta4M 4M + 5.4 Circular Orbit CCAS	Aluminum 7075 T73	Thin films	NiH2 CPV
53	694.93	298.4	4.34	5.9	0.105	1010.8	Atlas5 V-552 Circular Single Burn	Aluminum 6061 T6	Thin films	Li Ion
54	1016.35	298.4	4.26	5.9	0.105	565.1	Atlas5 V-552 Circular Single Burn	Aluminum 6061 T6	Thin films	Li Ion
55	609.42	295.6	4.26	5.9	0.106	565.1	Atlas5 V-552 Circular Single Burn	Aluminum 6061 T6	Thin films	NaS
56	609.42	295.6	4.26	5.54	0.106	565.1	Atlas5 V-552 Circular Single Burn	Aluminum 6061 T6	Thin films	NiH2 CPV
107	609.42	298.4	4.6	5.9	0.108	565.1	Atlas5 V-552 Circular Double Burn	Aluminum 7075 T73	GaAs dual junction	NiH2 CPV
108	609.42	295.6	4.26	5.54	0.125	565.1	Atlas5 V-552 Circular Double Burn	Aluminum 7075 T73	Thin films	NiH2 CPV
109	609.42	298.4	4.48	5.85	0.125	565.1	Atlas5 V-552 Circular Double Burn	Aluminum 7075 T73	Thin films	NiH2 CPV
110	609.42	295.6	4.48	5.65	0.127	565.1	Atlas5 V-552 Circular Double Burn	Aluminum 7075 T73	Thin films	NiH2 IPV
111	609.42	298.4	4.6	5.9	0.127	565.1	Atlas5 V-552 Circular Double Burn	Aluminum 7075 T73	GaAs dual junction	NiH2 CPV
113	406.96	298.4	4.26	5.9	0.105	1854.2	Ariane5ES LEO i48	Aluminum 6061 T6	Thin films	Li Ion
114	609.42	295.6	4.26	5.54	0.105	565.1	Ariane5ES LEO i48	Aluminum 6061 T6	Thin films	NaS
115	681.44	298.4	4.26	5.9	0.105	565.1	Ariane5ES LEO i48	Aluminum 6061 T6	Thin films	Li Ion
116	609.42	295.6	4.26	5.9	0.105	1854.2	Ariane5ES LEO i48	Aluminum 6061 T6	Thin films	NaS
117	609.42	295.6	4.26	5.99	0.105	853.4	Ariane5ES LEO i48	Aluminum 6061 T6	Thin films	Li Ion
118	921.53	298.4	4.31	5.68	0.106	565.1	Ariane5ES LEO i48	Aluminum 6061 T6	Thin films	Li Ion
119	913.2	298.4	4.31	5.9	0.105	1854.2	Ariane5ES LEO i48	Aluminum 6061 T6	Thin films	Li Ion
120	558.27	295.6	4.26	5.54	0.106	565.1	Ariane5ES LEO i60	Aluminum 7075 T73	Thin films	NaS
121	694.93	298.4	4.34	5.9	0.105	1010.8	Ariane5ES LEO i60	Aluminum 6061 T6	Thin films	Li Ion
122	609.42	298.4	4.48	5.86	0.106	565.1	Ariane5ES LEO i48	Aluminum 7075 T73	Thin films	Li Ion

Table 8.9: Some of the non-dominated design points: objective functions and constraints (subcase 3).

ID.	Cost (M$)	Mass (kg)	Volume (m^3)	Energy (KWh)	con. 1	con. 2	con. 3	con. 4
1	75	922.07	80.62	97.03	-0.61	-1.23	-0.59	-0.1
2	75	881.84	79.81	96.99	-0.59	-1.23	-0.59	-0.1
3	75	877.88	79.53	38.78	-0.59	-0.53	-0.59	-0.16
4	75	860.97	80.61	38.78	-0.57	-0.52	-0.59	-0.15
5	75	860.97	80.61	38.78	-0.57	-0.52	-0.59	-0.15
6	75	810.91	82.92	38.82	-0.54	-0.29	-0.59	-0.18
7	75	806.61	85.3	97.41	-0.54	-1.23	-0.59	-0.1
8	75	777.41	85.22	96.84	-0.52	-1.23	-1.08	-0.06
9	75	771.76	83.67	60.57	-0.51	-1.02	-0.97	-0.18
10	75	769.81	78.4	38.74	-0.51	-0.56	-0.98	-0.24
52	80	739.09	79.18	38.57	-0.49	-0.77	-1.13	-0.64
53	85	2624.06	86.54	63.74	-1.75	-1.07	-0.58	-0.07
54	85	2583.81	83.46	38.85	-1.72	-0.24	-0.59	-0.12
55	85	2583.19	83.45	38.79	-1.72	-0.27	-0.59	-0.16
56	85	2543.62	78.14	38.73	-1.7	-0.52	-0.59	-0.16
107	90	2293.51	96.93	38.72	-1.53	-0.48	-1.08	-0.04
108	90	2292.93	78.05	38.58	-1.53	-0.76	-1.13	-0.65
109	90	2114.56	91.12	38.63	-1.41	-0.64	-1.12	-0.51
110	90	2005.59	87.8	38.55	-1.34	-0.79	-1.12	-0.58
111	90	1895.14	96.82	38.6	-1.26	-0.69	-1.12	-0.5
113	120	2920.84	83.46	97.3	-1.95	-1.23	-0.59	-0.11
114	120	2885.38	78.41	38.74	-1.92	-0.5	-0.59	-0.12
115	120	2842.2	83.46	38.81	-1.89	-0.24	-0.59	-0.12
116	120	2801.16	83.46	97.36	-1.87	-1.23	-0.59	-0.12
117	120	2779.36	84.75	55.58	-1.85	-0.7	-0.59	-0.12
118	120	2745.18	82.15	38.78	-1.83	-0.44	-0.59	-0.13
119	120	2727.23	85.29	97.41	-1.82	-1.23	-0.59	-0.09
120	120	2716.72	78.4	38.7	-1.81	-0.55	-1.09	-0.2
121	120	2652.96	86.54	63.74	-1.77	-1.07	-0.58	-0.07
122	120	2650.89	91.58	38.74	-1.77	-0.43	-1.08	-0.07

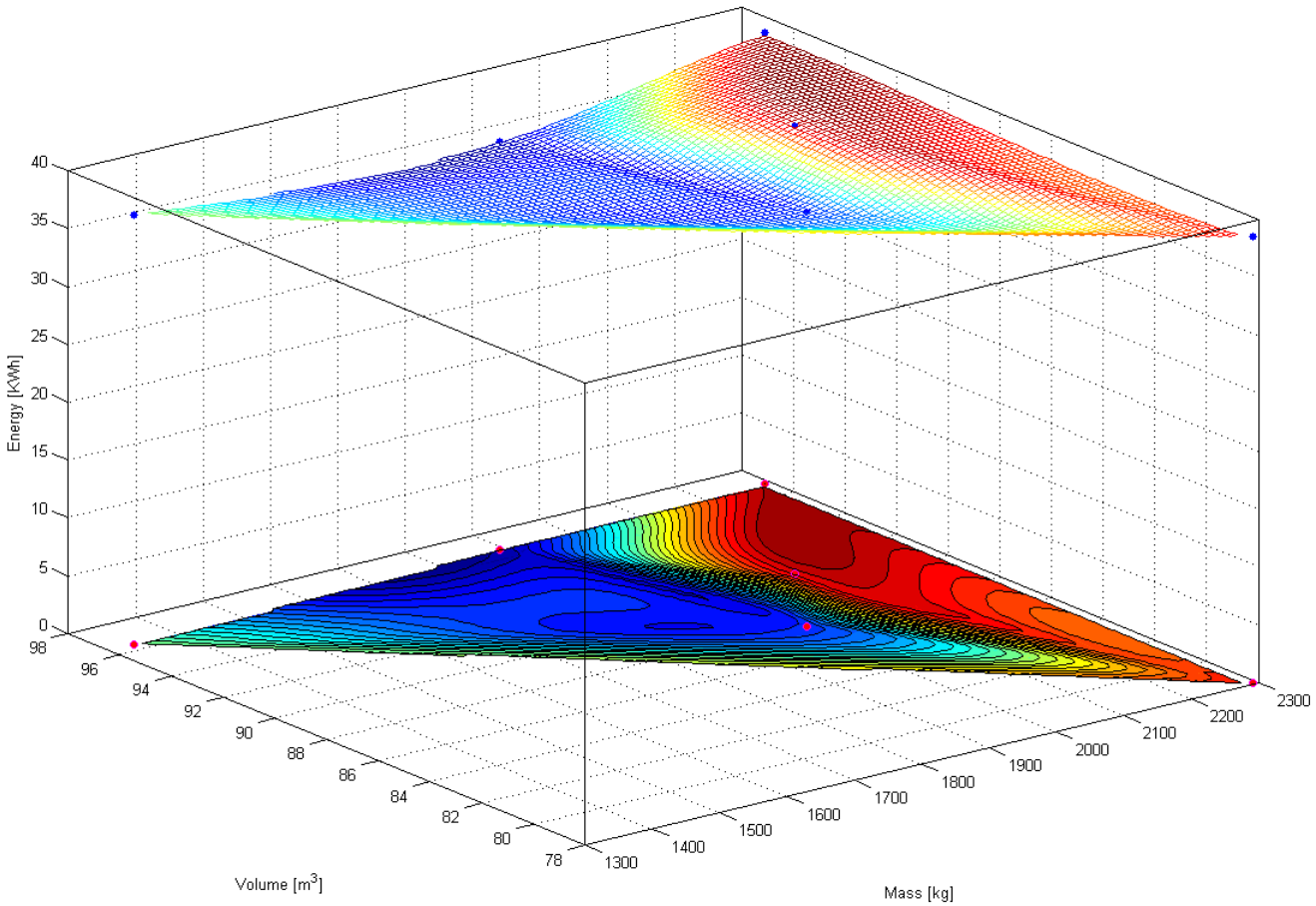

Figure 8.22: Pareto front corresponding to 90 M$ launch cost.

Table 8.10: Parameters of the MOGA method used for the iterations cycle of subcase 4.

Parameter	Value
initialization type	unique random
crossover type	multi point binary
crossover rate	0.65
mutation type	replace uniform
mutation rate	0.25
fitness type	domination count
replacement type	unique roulette wheel
shrinkage percentage	0.8
percent change (convergence)	0.05

Iterations history

The results related to the objective functions and constraints are reported in the figures 8.24 and 8.25. In particular the quantities are reported with respect to the iteration number and a fitting curve is also introduced to give an approximation of the overall evolution during the iterations cycle.

Pareto fronts

The Pareto fronts related to the non-dominated design points are also reported in figures 8.26, 8.27,8.28 and 8.29. In particular since four objective functions are available it is difficult to plot them in a readable way. For this reason three of them are reported in the 3d plots (mass, volume and energy) with respect to a specific launch cost. The launch costs corresponding to the non-dominated solutions identified by the

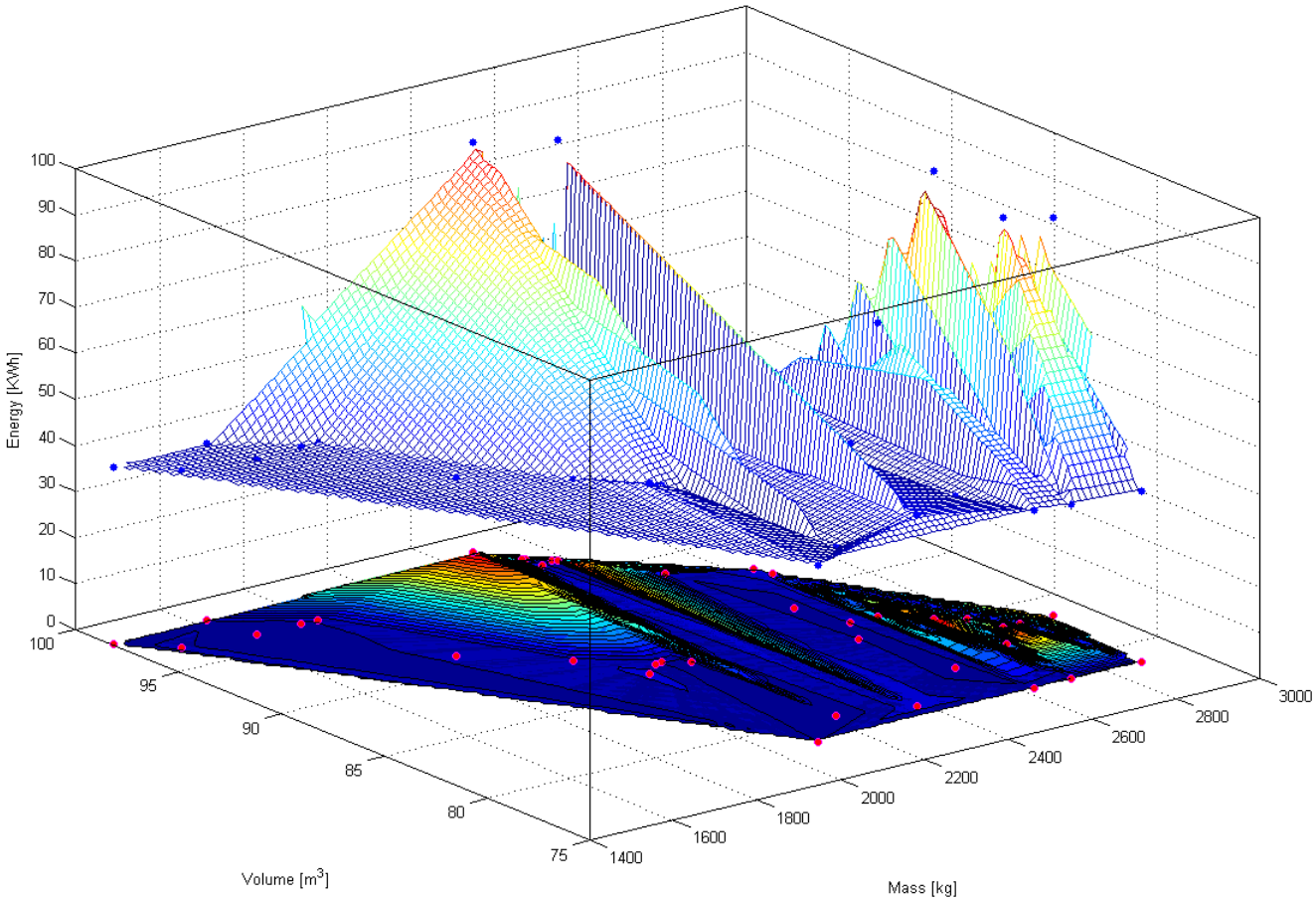

Figure 8.23: Pareto front corresponding to 120 M$ launch cost.

algorithm can in fact be used as a parameter in such representation (the discrete range of launch cost helps to pursue such data representation). In this way each cost has its corresponding Pareto front reported in three dimensions.

Optimal design points summary

Some of the non-dominated solutions identified by the solving algorithm are reported in tables 8.11 and 8.12.

8.4.5 Subcase 5

Optimization parameters

The optimization parameters used for the current subcase are reported in table 8.13

Table 8.11: Some of the non-dominated design points: design variables (subcase 4).

ID.	H (Km)	Sp (sec)	D (m)	L (m)	Thk (m)	P (W)	La	Al	C	B
1	453.77	290.2	4.35	5.88	0.103	849.4	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 7075 T73	GaInP dual junction	NaS
2	453.77	290.2	4.58	5.99	0.111	849.4	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 7075 T73	Thin films	NaS
3	453.77	286.2	4.58	5.76	0.111	575.9	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 6061 T6	Silicon	NiMH
4	648.91	296.2	4.58	5.95	0.111	874.2	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 7075 T73	Thin films	NaS
5	555.55	296.2	4.58	5.95	0.111	874.2	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 7075 T73	Silicon	NaS
6	453.77	296.2	4.29	5.97	0.103	849.4	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 6061 T6	Silicon	NaS
7	453.77	299.2	4.29	5.97	0.103	659.3	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 2219 T851	Silicon	NaS
8	453.77	296.2	4.45	5.83	0.103	735.1	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 7075 T73	GaAs dual junction	NaS
9	434.36	290.2	4.31	5.87	0.103	659.3	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 7075 T73	GaAs dual junction	NaS
10	1059.93	296.2	4.31	5.87	0.103	575.9	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 2219 T851	GaAs dual junction	NaS
86	495.71	299.2	4.31	5.74	0.107	659.3	Atlas5 V-552 Circular Double Burn	Aluminum 6061 T6	Thin films	NaS
87	495.71	296.2	4.32	5.74	0.107	659.3	Atlas5 V-552 Circular Double Burn	Aluminum 6061 T6	GaInP dual junction	Li Ion
88	495.71	299.2	4.31	5.74	0.107	642.2	Atlas5 V-552 Circular Double Burn	Aluminum 2219 T851	Thin films	NaS
89	453.77	299.2	4.45	5.97	0.107	1110.5	Atlas5 V-552 Circular Double Burn	Aluminum 6061 T6	Thin films	NaS
90	453.77	296.2	4.43	5.74	0.107	575.9	Atlas5 V-552 Circular Double Burn	Aluminum 6061 T6	GaInP dual junction	Li Ion
91	453.77	296.2	4.21	5.63	0.111	575.9	Atlas5 V-552 Circular Double Burn	Aluminum 2219 T851	GaInP dual junction	Li Ion
92	495.71	299.2	4.45	5.74	0.112	735.1	Atlas5 V-552 Circular Double Burn	Aluminum 6061 T6	GaInP dual junction	NaS
93	453.77	296.2	4.45	5.97	0.107	659.3	Atlas5 V-552 Circular Double Burn	Aluminum 6061 T6	Thin films	NaS
94	453.77	296.2	4.31	5.74	0.112	575.9	Atlas5 V-552 Circular Double Burn	Aluminum 6061 T6	Silicon	Li Ion
95	495.71	299.2	4.58	5.74	0.111	659.3	Atlas5 V-552 Circular Double Burn	Aluminum 6061 T6	Thin films	NaS
125	434.36	299.2	4.31	5.97	0.107	659.3	Ariane5ES LEO i60	Aluminum 6061 T6	GaAs dual junction	NaS
126	453.77	299.2	4.41	5.97	0.107	659.3	Ariane5ES LEO i60	Aluminum 6061 T6	Thin films	NaS
127	453.77	299.2	4.31	5.84	0.111	659.3	Ariane5ES LEO i60	Aluminum 6061 T6	GaAs dual junction	Li Ion
128	495.71	296.2	4.32	5.88	0.111	594.8	Ariane5ES LEO i48	Aluminum 6061 T6	GaInP dual junction	Li Ion
129	495.71	296.2	4.45	5.88	0.112	659.3	Ariane5ES LEO i48	Aluminum 6061 T6	Thin films	Li Ion
130	453.77	299.2	4.45	5.97	0.107	659.3	Ariane5ES LEO i48	Aluminum 7075 T73	Silicon	NaS
131	453.77	299.2	4.58	5.8	0.111	575.9	Ariane5ES LEO i48	Aluminum 6061 T6	GaAs dual junction	NaS
132	453.77	299.2	4.58	5.97	0.107	659.3	Ariane5ES LEO i48	Aluminum 7075 T73	Silicon	NaS
133	464.67	299.2	4.58	5.99	0.111	874.2	Ariane5ES LEO i48	Aluminum 6061 T6	GaAs dual junction	Li Ion
134	434.36	299.2	4.45	5.88	0.119	567.3	Ariane5ES LEO i60	Aluminum 6061 T6	Thin films	NaS

Table 8.12: Some of the non-dominated design points: objective functions and constraints (subcase 4).

ID.	Cost (M$)	Mass (kg)	Volume (m^3)	Energy (KWh)	con. 1	con. 2	con. 3	con. 4
1	65	482.54	86.62	55.26	-0.32	-0.81	-1.08	-0.07
2	65	165.32	97.89	55.12	-0.11	-0.94	-1.09	-0.15
3	65	54.26	94.04	39.34	-0.04	-0.59	-0.59	-0.11
4	70	385.06	97.29	56.46	-0.26	-1.0	-1.09	-0.15
5	70	324.74	97.29	56.45	-0.22	-1.0	-1.09	-0.15
6	75	836.58	85.63	55.35	-0.56	-0.69	-0.58	-0.06
7	75	780.52	85.63	44.48	-0.52	-0.4	-0.96	-0.12
8	75	725.33	89.81	48.84	-0.48	-0.68	-1.07	-0.0
9	75	707.51	85.16	44.46	-0.47	-0.46	-1.08	-0.08
10	75	675.55	84.99	39.49	-0.45	-0.33	-0.96	-0.11
86	90	2824.39	82.95	44.43	-1.88	-0.58	-0.59	-0.16
87	90	2724.45	83.45	44.42	-1.82	-0.59	-0.59	-0.15
88	90	2724.3	82.95	43.37	-1.82	-0.6	-0.98	-0.22
89	90	2676.18	91.95	68.48	-1.78	-1.22	-0.59	-0.07
90	90	2651.17	87.75	39.38	-1.77	-0.49	-0.59	-0.08
91	90	2638.8	77.68	39.33	-1.76	-0.56	-0.99	-0.38
92	90	2637.52	88.41	48.75	-1.76	-0.83	-0.6	-0.18
93	90	2605.31	92.16	44.46	-1.74	-0.49	-0.58	-0.07
94	90	2592.92	83.26	39.37	-1.73	-0.49	-0.6	-0.26
95	90	2589.96	93.91	44.34	-1.73	-0.75	-0.59	-0.09
125	120	2727.89	86.63	44.48	-1.82	-0.43	-0.59	-0.15
126	120	2723.74	90.35	44.46	-1.82	-0.47	-0.59	-0.1
127	120	2681.87	84.36	44.42	-1.79	-0.56	-0.6	-0.26
128	120	2673.44	85.46	40.56	-1.78	-0.42	-0.6	-0.25
129	120	2611.19	90.57	44.41	-1.74	-0.6	-0.6	-0.18
130	120	2608.79	92.16	44.43	-1.74	-0.52	-1.08	-0.11
131	120	2576.05	94.87	39.35	-1.72	-0.56	-0.59	-0.09
132	120	2528.28	97.66	44.41	-1.69	-0.58	-1.08	-0.04
133	120	2523.62	97.89	56.5	-1.68	-0.95	-0.59	-0.09
134	120	2508.42	90.53	38.81	-1.67	-0.53	-0.61	-0.37

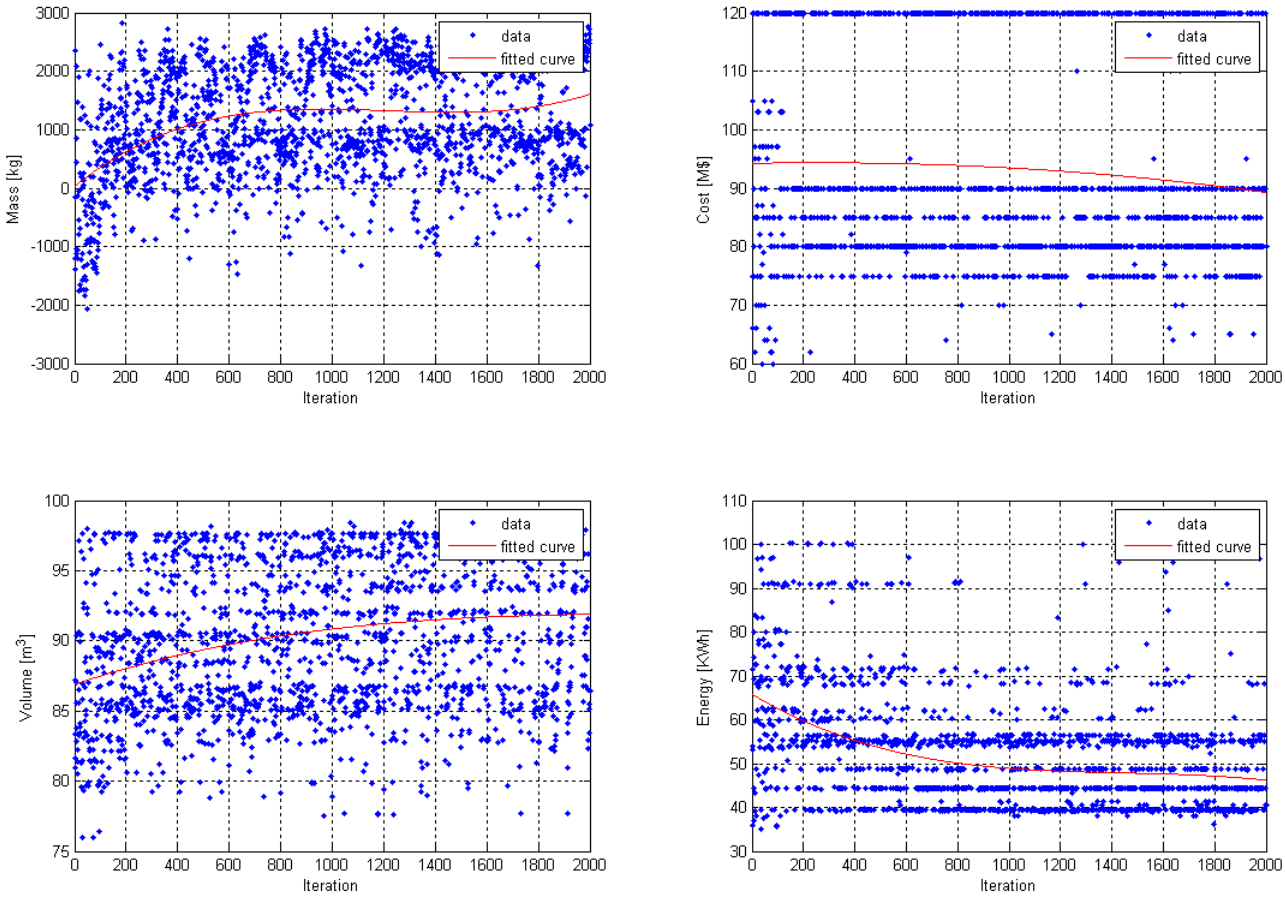

Figure 8.24: Objective functions.

Table 8.13: Parameters of the MOGA method used for the iterations cycle of subcase 5.

Parameter	Value
initialization type	unique random
crossover type	multi point binary
crossover rate	0.7
mutation type	offset normal
mutation rate	0.2
fitness type	layer rank
replacement type	below limit
shrinkage percentage	0.85
percent change (convergence)	0.05

Iterations history

The results related to the objective functions and constraints are reported in the figures 8.30 and 8.31. In particular the quantities are reported with respect to the iteration number and a fitting curve is also introduced to give an approximation of the overall evolution during the iterations cycle.

Pareto fronts

The Pareto fronts related to the non-dominated design points are also reported in figures 8.32, 8.33, 8.34,8.35 and 8.36. In particular since four objective functions are available it is difficult to plot them in a readable way. For this reason three of them are reported in the 3d plots (mass, volume and energy) with

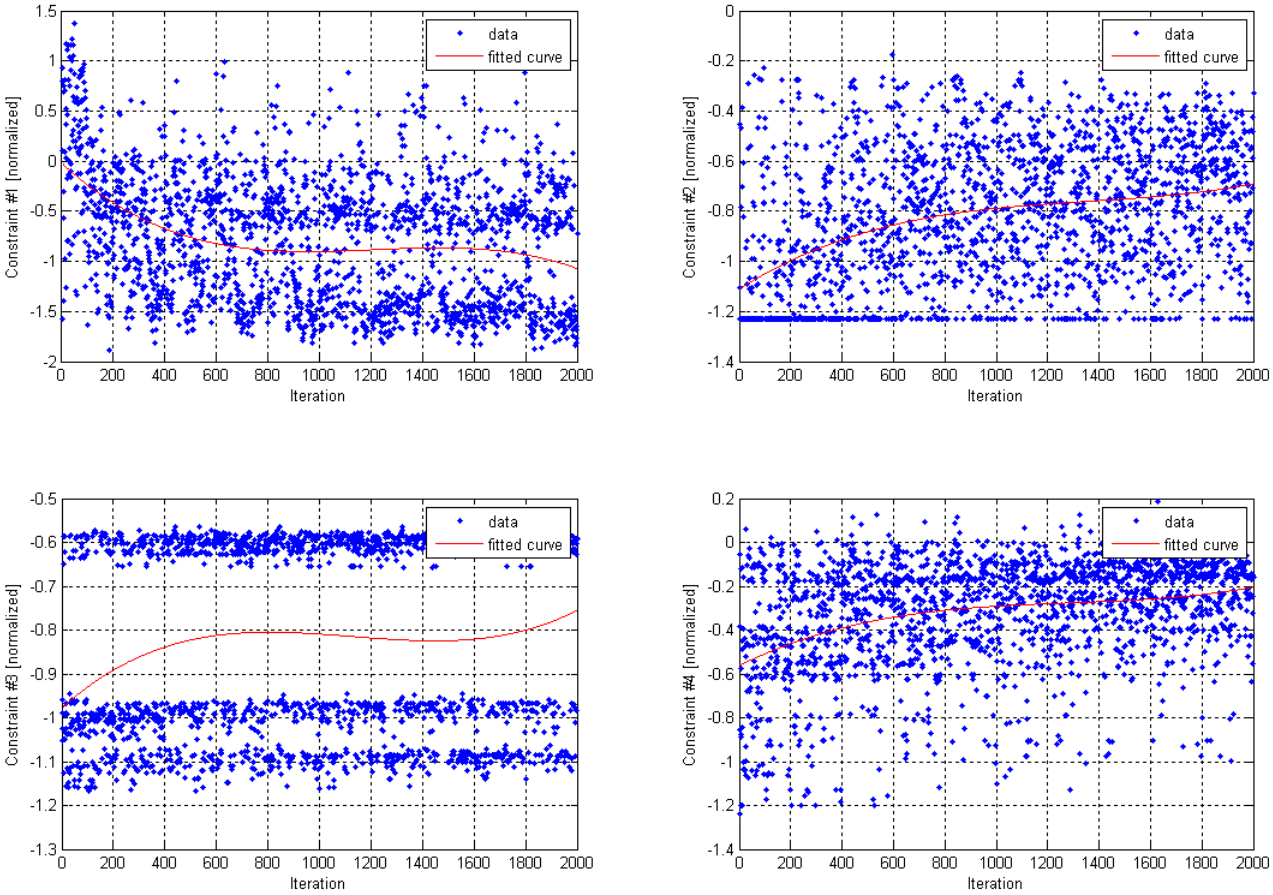

Figure 8.25: Constraints.

respect to a specific launch cost. The launch costs corresponding to the non-dominated solutions identified by the algorithm can in fact be used as a parameter in such representation (the discrete range of launch cost helps to pursue such data representation). In this way each cost has its corresponding Pareto front reported in three dimensions.

Optimal design points summary

Some of the non-dominated solutions identified by the solving algorithm are reported in tables 8.14 and 8.15.

8.5 Considerations about the results

The results reported in the previous sections are directly obtained through the support of the developed framework, exploiting the capabilities provided by DAKOTA. For the sake of clarity the graphical representations of the data (in particular the iteration histories and pareto fronts) have been realized within Matlab environment but the same capabilities can potentially be implemented within the same platform (such feature is currently under evaluation within the development roadmap). These results showed how and in which manner a model based approach can further enhance the trade-off analyses or optimization cycles. The reference case has been used to assess the capability to give more consistency to the overall design process, ensuring a more seamless connection between the modeling and analysis environments.
The ranges of the considered design variables as well as the quantities considered as the objective functions have been directly extracted from the information available from the web-based infrastructure. These results showed how a consistent way of data exchange can help to formalize the management of complex system, reducing the error-prone process of model transformation for example.

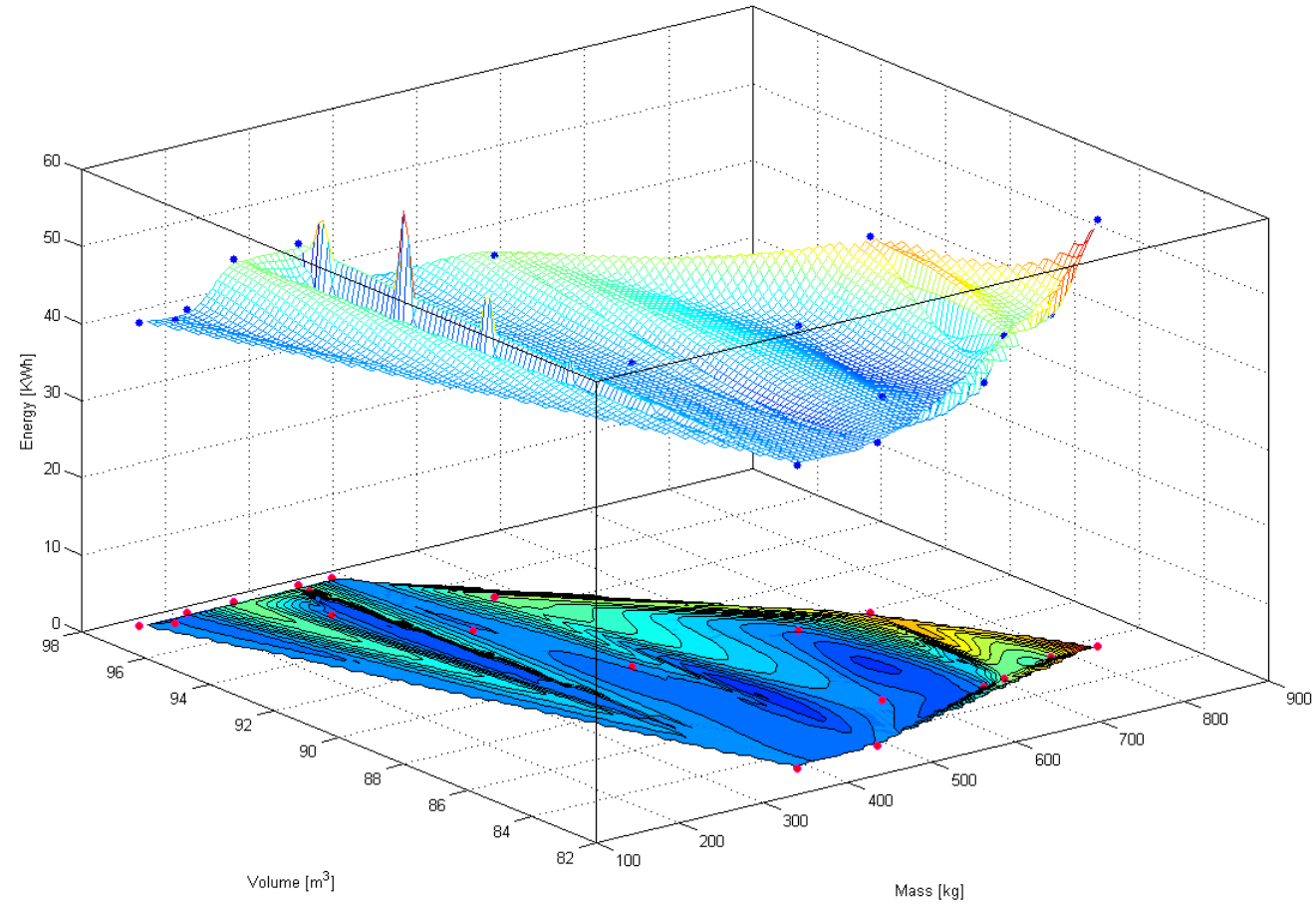

Figure 8.26: Pareto front corresponding to 75 M$ launch cost.

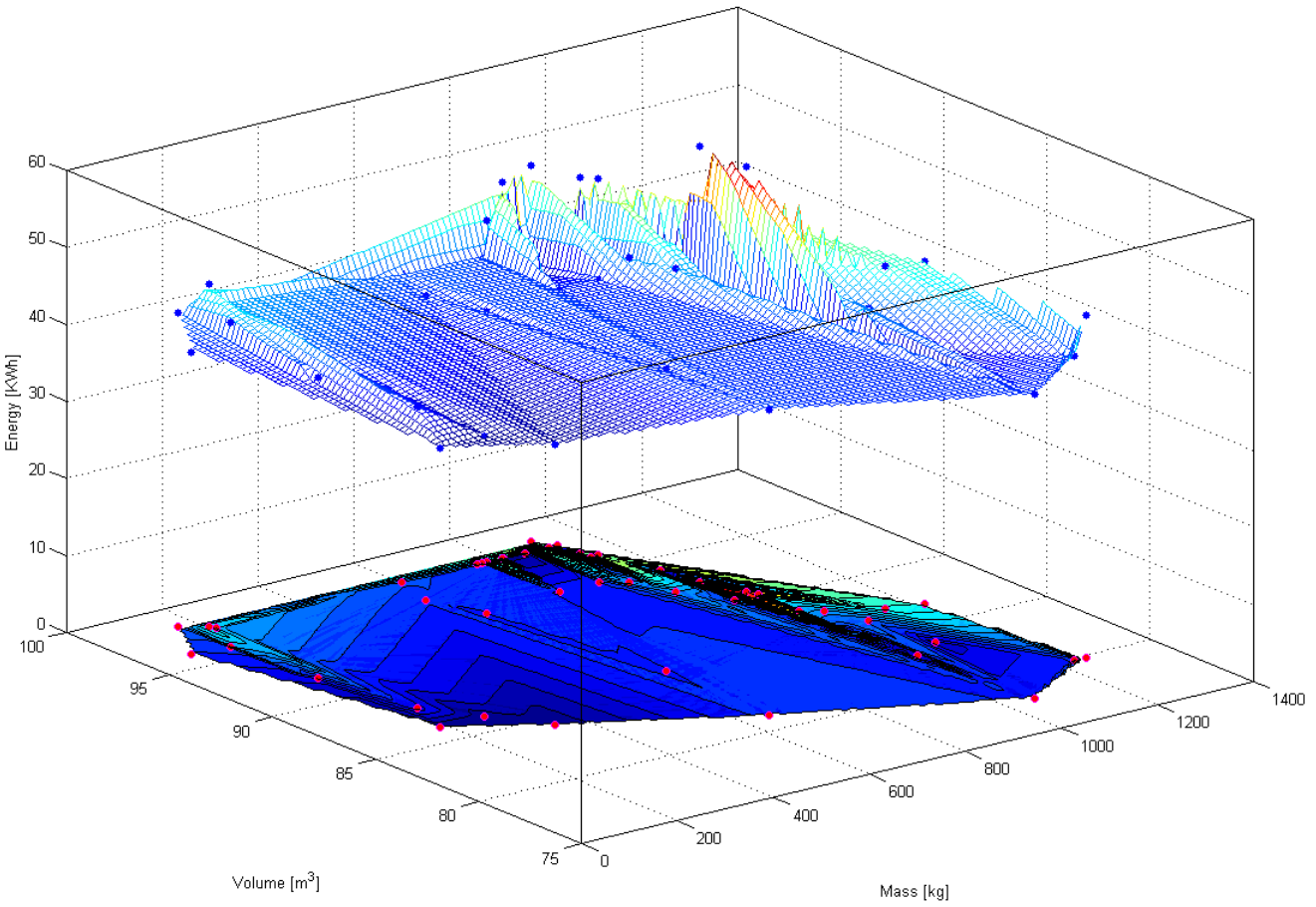

Figure 8.27: Pareto front corresponding to 85 M$ launch cost.

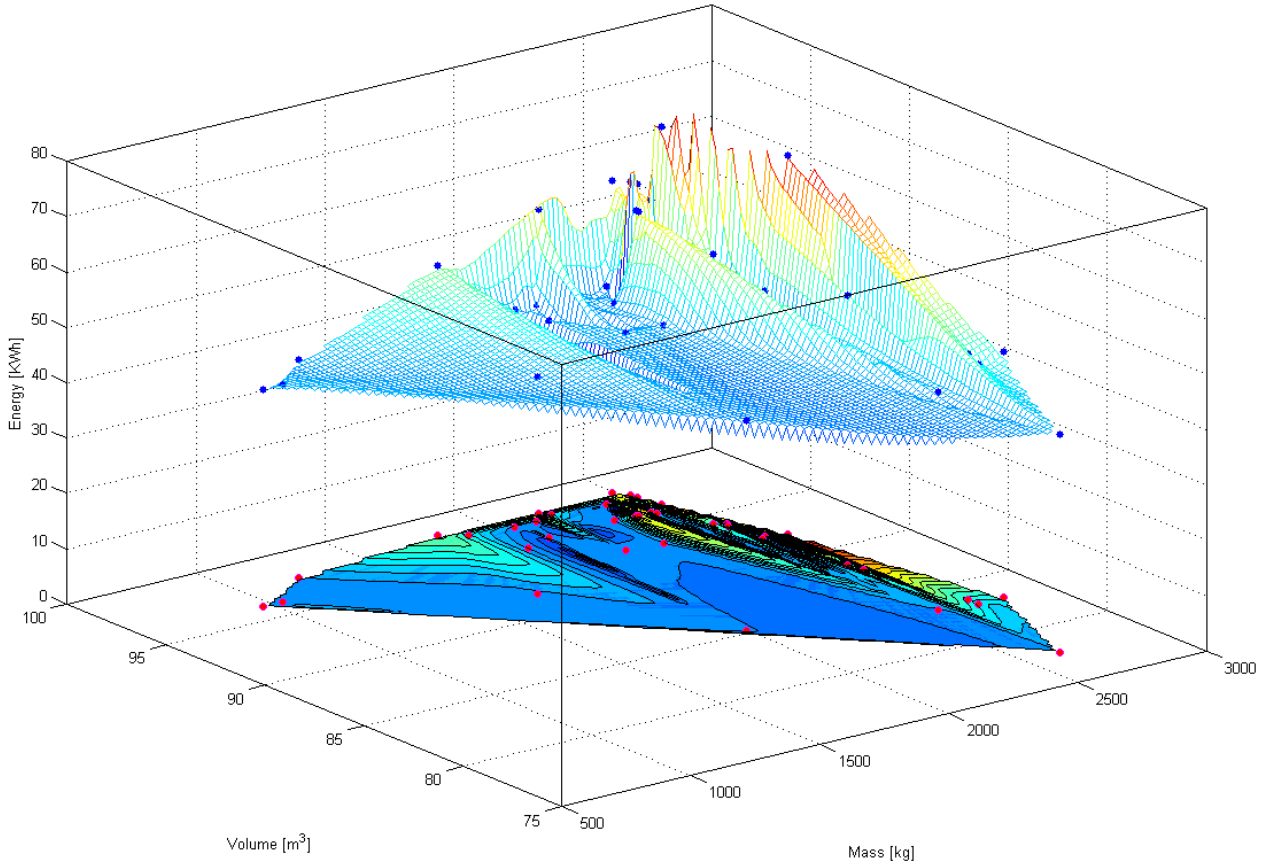

Figure 8.28: Pareto front corresponding to 90 M$ launch cost.

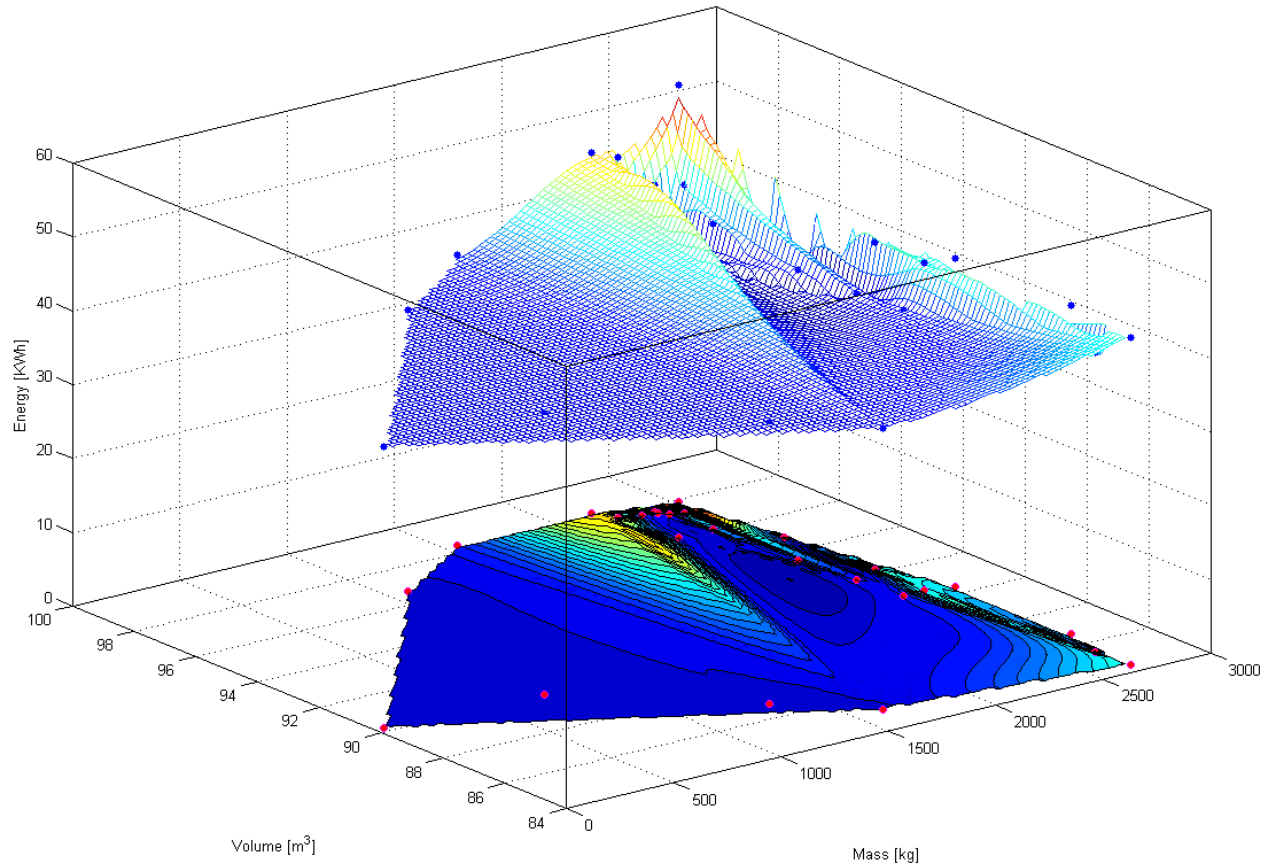

Figure 8.29: Pareto front corresponding to 120 M$ launch cost.

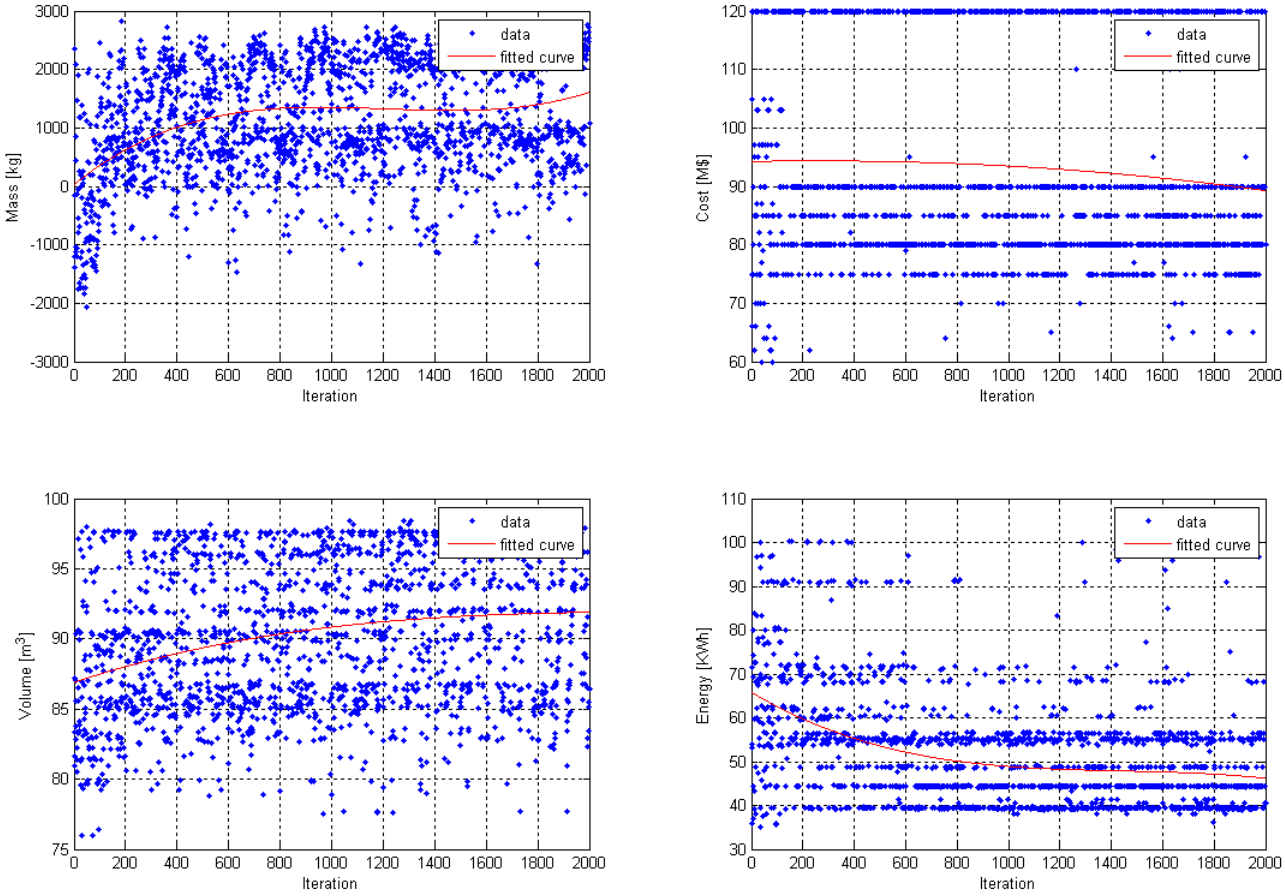

Figure 8.30: Objective functions.

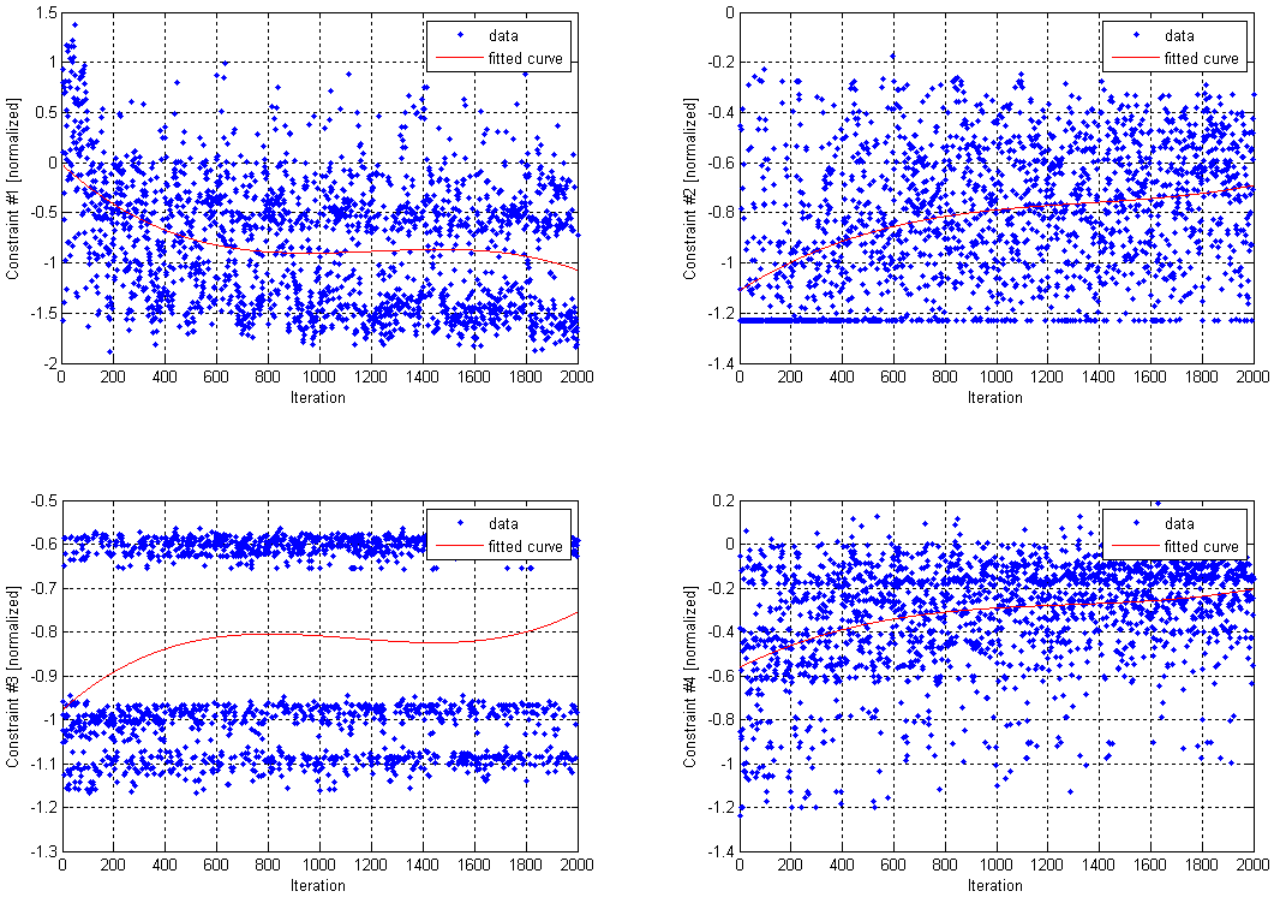

Figure 8.31: Constraints.

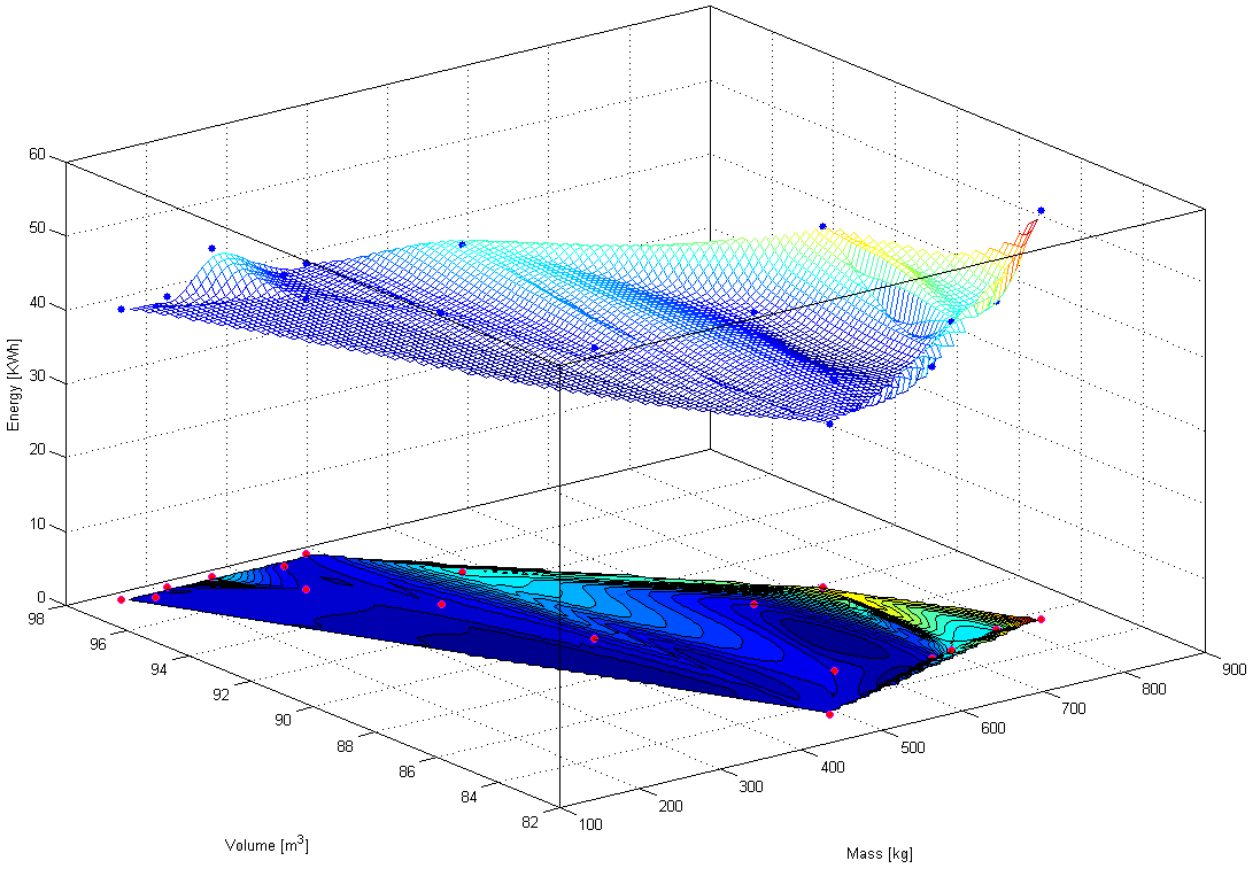

Figure 8.32: Pareto front corresponding to 75 M$ launch cost.

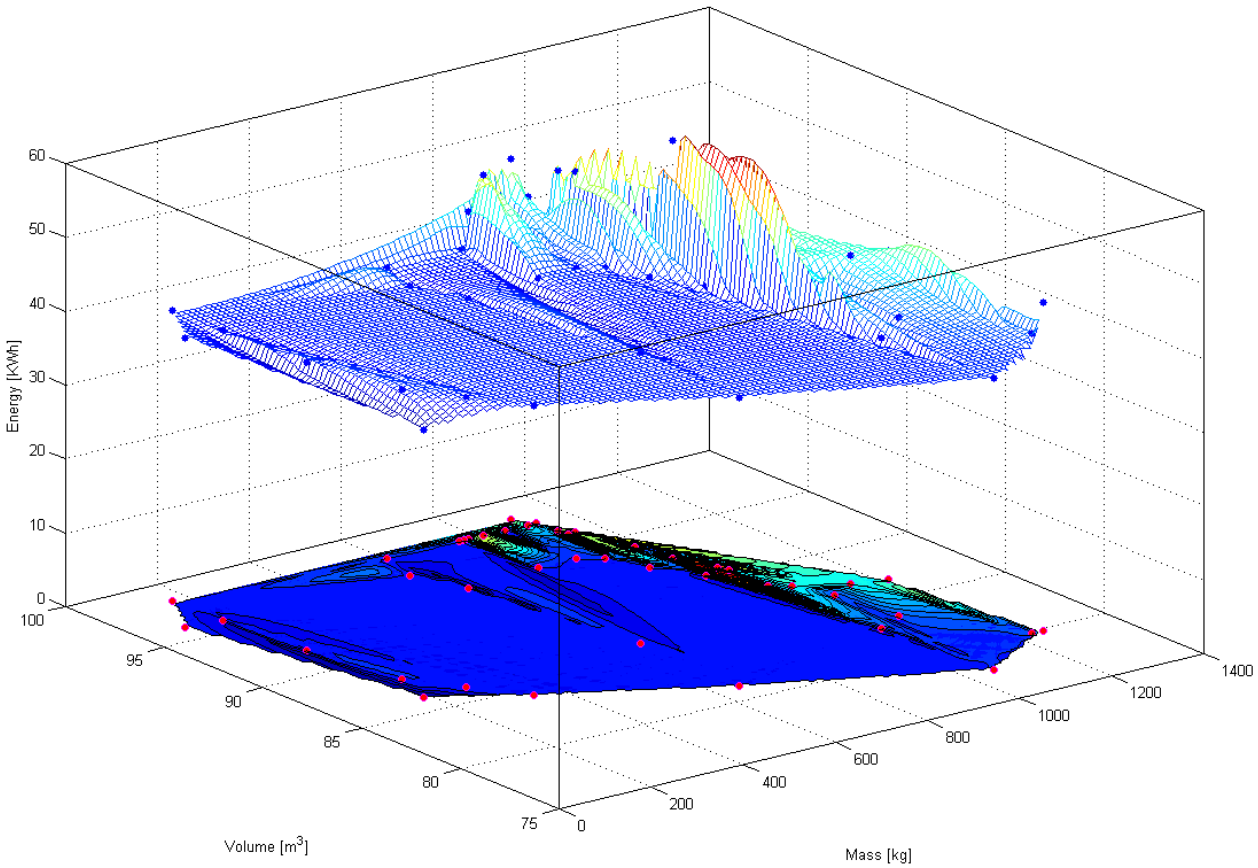

Figure 8.33: Pareto front corresponding to 80 M$ launch cost.

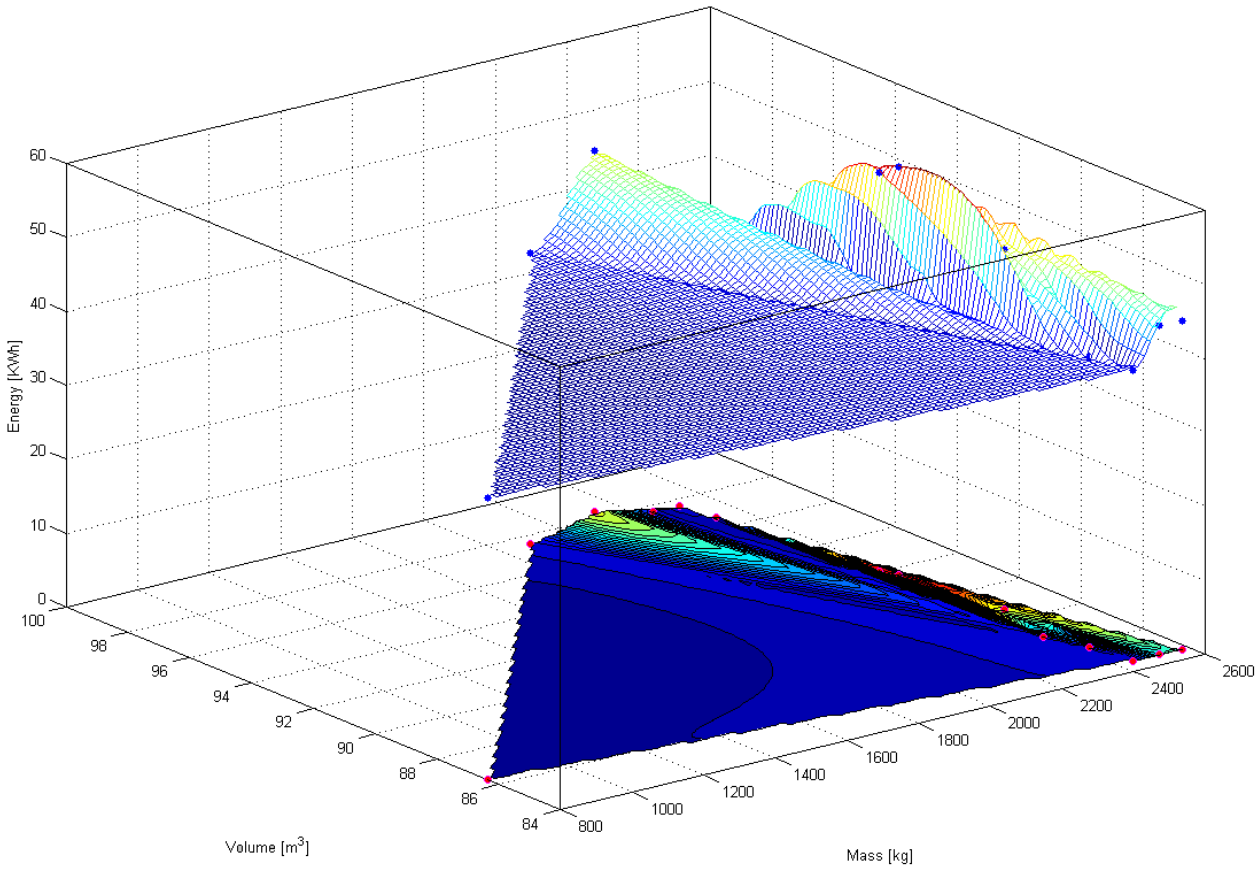

Figure 8.34: Pareto front corresponding to 85 M$ launch cost.

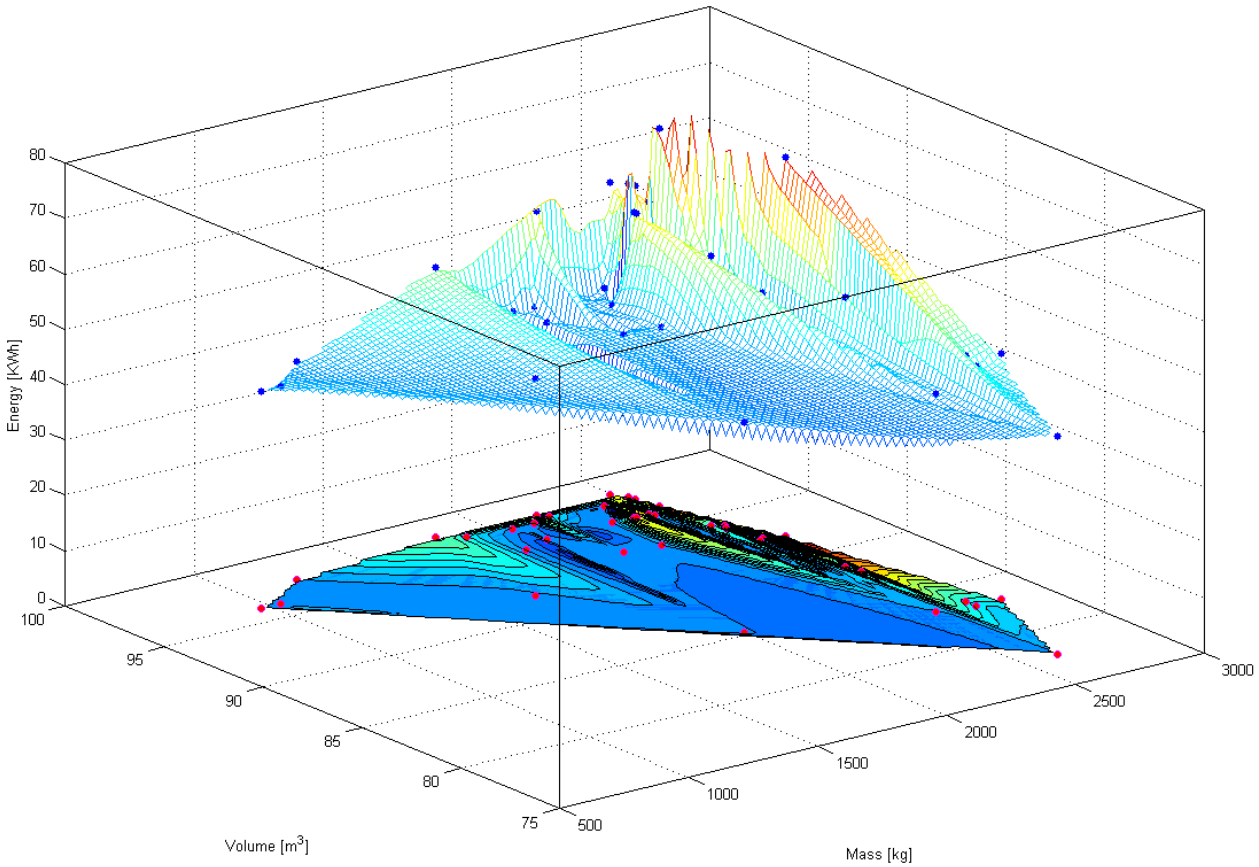

Figure 8.35: Pareto front corresponding to 90 M$ launch cost.

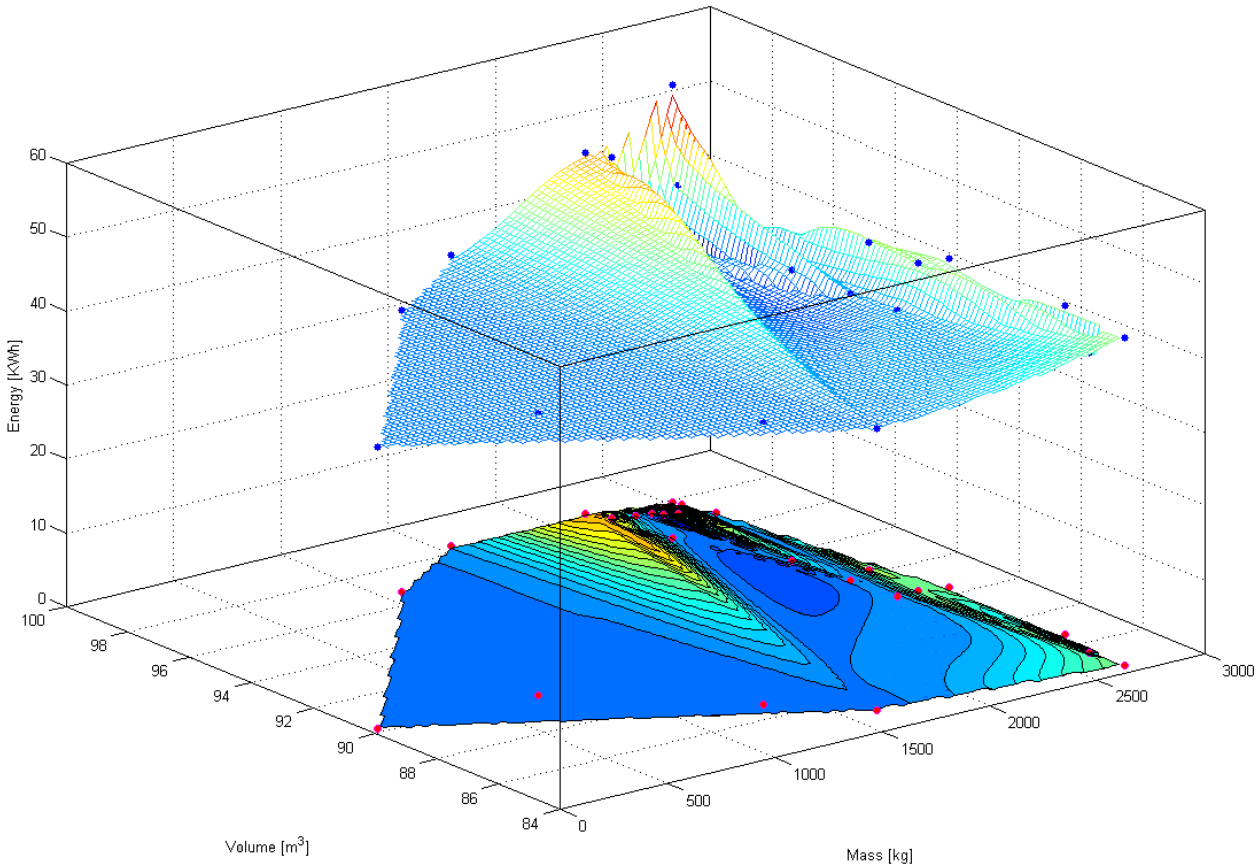

Figure 8.36: Pareto front corresponding to 120 M$ launch cost.

Table 8.14: Some of the non-dominated design points: design variables (subcase 5).

ID.	H (Km)	Sp (sec)	D (m)	L (m)	Thk (m)	P (W)	La	Al	C	B
1	453.77	290.2	4.35	5.88	0.103	849.4	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 7075 T73	GaInP dual junction	NaS
2	453.77	290.2	4.58	5.99	0.111	849.4	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 7075 T73	Thin films	NaS
3	453.77	286.2	4.58	5.76	0.111	575.9	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 6061 T6	Silicon	NiMH
4	648.91	296.2	4.58	5.95	0.111	874.2	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 7075 T73	Thin films	NaS
5	555.55	296.2	4.58	5.95	0.111	874.2	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 7075 T73	Silicon	NaS
6	453.77	296.2	4.29	5.97	0.103	849.4	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 6061 T6	Silicon	NaS
7	453.77	299.2	4.29	5.97	0.103	659.3	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 2219 T851	Silicon	NaS
8	453.77	296.2	4.45	5.83	0.103	735.1	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 7075 T73	GaAs dual junction	NaS
9	434.36	290.2	4.31	5.87	0.103	659.3	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 7075 T73	GaAs dual junction	NaS
10	1059.93	296.2	4.31	5.87	0.103	575.9	Delta4M 4M + 4.2 Circular Orbit CCAS	Aluminum 2219 T851	GaAs dual junction	NaS
72	453.77	299.2	4.35	5.74	0.111	659.3	Atlas5 V-552 Circular Single Burn	Aluminum 6061 T6	Thin films	NaS
73	1006.16	299.2	4.35	5.74	0.107	659.3	Atlas5 V-552 Circular Single Burn	Aluminum 6061 T6	GaInP dual junction	NaS
74	453.77	299.2	4.35	5.74	0.111	575.9	Atlas5 V-552 Circular Single Burn	Aluminum 7075 T73	Silicon	NaS
75	453.77	299.2	4.45	5.76	0.112	735.1	Atlas5 V-552 Circular Single Burn	Aluminum 6061 T6	Silicon	NaS
76	434.36	299.2	4.31	5.91	0.112	575.9	Atlas5 V-552 Circular Single Burn	Aluminum 2219 T851	GaInP dual junction	Li Ion
86	495.71	299.2	4.31	5.74	0.107	659.3	Atlas5 V-552 Circular Double Burn	Aluminum 6061 T6	Thin films	NaS
87	495.71	296.2	4.32	5.74	0.107	659.3	Atlas5 V-552 Circular Double Burn	Aluminum 6061 T6	GaInP dual junction	Li Ion
88	495.71	299.2	4.31	5.74	0.107	642.2	Atlas5 V-552 Circular Double Burn	Aluminum 2219 T851	Thin films	NaS
89	453.77	299.2	4.45	5.97	0.107	1110.5	Atlas5 V-552 Circular Double Burn	Aluminum 6061 T6	Thin films	NaS
90	453.77	296.2	4.43	5.74	0.107	575.9	Atlas5 V-552 Circular Double Burn	Aluminum 6061 T6	GaInP dual junction	Li Ion
125	434.36	299.2	4.31	5.97	0.107	659.3	Ariane5ES LEO i60	Aluminum 6061 T6	GaAs dual junction	NaS
126	453.77	299.2	4.41	5.97	0.107	659.3	Ariane5ES LEO i60	Aluminum 6061 T6	Thin films	NaS
127	453.77	299.2	4.31	5.84	0.111	659.3	Ariane5ES LEO i60	Aluminum 6061 T6	GaAs dual junction	Li Ion
128	495.71	296.2	4.32	5.88	0.111	594.8	Ariane5ES LEO i48	Aluminum 6061 T6	GaInP dual junction	Li Ion
129	495.71	296.2	4.45	5.88	0.112	659.3	Ariane5ES LEO i48	Aluminum 6061 T6	Thin films	Li Ion
130	453.77	299.2	4.45	5.97	0.107	659.3	Ariane5ES LEO i48	Aluminum 7075 T73	Silicon	NaS
131	453.77	299.2	4.58	5.8	0.111	575.9	Ariane5ES LEO i48	Aluminum 6061 T6	GaAs dual junction	NaS
132	453.77	299.2	4.58	5.97	0.107	659.3	Ariane5ES LEO i48	Aluminum 7075 T73	Silicon	NaS
133	464.67	299.2	4.58	5.99	0.111	874.2	Ariane5ES LEO i48	Aluminum 6061 T6	GaAs dual junction	Li Ion
134	434.36	299.2	4.45	5.88	0.119	567.3	Ariane5ES LEO i60	Aluminum 6061 T6	Thin films	NaS

Table 8.15: Some of the non-dominated design points: objective functions and constraints (subcase 5).

ID.	Cost (M$)	Mass (kg)	Volume (m^3)	Energy (KWh)	con. 1	con. 2	con. 3	con. 4
1	65	482.54	86.62	55.26	-0.32	-0.81	-1.08	-0.07
2	65	165.32	97.89	55.12	-0.11	-0.94	-1.09	-0.15
3	65	54.26	94.04	39.34	-0.04	-0.59	-0.59	-0.11
4	70	385.06	97.29	56.46	-0.26	-1.0	-1.09	-0.15
5	70	324.74	97.29	56.45	-0.22	-1.0	-1.09	-0.15
6	75	836.58	85.63	55.35	-0.56	-0.69	-0.58	-0.06
7	75	780.52	85.63	44.48	-0.52	-0.4	-0.96	-0.12
8	75	725.33	89.81	48.84	-0.48	-0.68	-1.07	-0.0
9	75	707.51	85.16	44.46	-0.47	-0.46	-1.08	-0.08
10	75	675.55	84.99	39.49	-0.45	-0.33	-0.96	-0.11
72	85	2581.84	84.51	44.38	-1.72	-0.65	-0.6	-0.23
73	85	2518.54	84.53	44.48	-1.68	-0.6	-0.59	-0.14
74	85	2443.69	84.51	39.35	-1.63	-0.53	-1.1	-0.27
75	85	2438.71	88.61	48.75	-1.63	-0.82	-0.6	-0.18
76	85	2424.97	85.69	39.37	-1.62	-0.43	-0.98	-0.33
86	90	2824.39	82.95	44.43	-1.88	-0.58	-0.59	-0.16
87	90	2724.45	83.45	44.42	-1.82	-0.59	-0.59	-0.15
88	90	2724.3	82.95	43.37	-1.82	-0.6	-0.98	-0.22
89	90	2676.18	91.95	68.48	-1.78	-1.22	-0.59	-0.07
90	90	2651.17	87.75	39.38	-1.77	-0.49	-0.59	-0.08
125	120	2727.89	86.63	44.48	-1.82	-0.43	-0.59	-0.15
126	120	2723.74	90.35	44.46	-1.82	-0.47	-0.59	-0.1
127	120	2681.87	84.36	44.42	-1.79	-0.56	-0.6	-0.26
128	120	2673.44	85.46	40.56	-1.78	-0.42	-0.6	-0.25
129	120	2611.19	90.57	44.41	-1.74	-0.6	-0.6	-0.18
130	120	2608.79	92.16	44.43	-1.74	-0.52	-1.08	-0.11
131	120	2576.05	94.87	39.35	-1.72	-0.56	-0.59	-0.09
132	120	2528.28	97.66	44.41	-1.69	-0.58	-1.08	-0.04
133	120	2523.62	97.89	56.5	-1.68	-0.95	-0.59	-0.09
134	120	2508.42	90.53	38.81	-1.67	-0.53	-0.61	-0.37

With respect to the reference case, the optimization capabilities of DAKOTA allowed to investigate a wide set of possible mission solutions and design choices, identifying groups of non-dominated configurations. In particular different Pareto fronts have been generated from the simulation scenarios considered in the reference scenario and such information is directly correlated with the data available from the system model. Such results can represent a useful instrument to iterate over the design process, taking into account solutions that have not been considered and improving the overall performances of the system. Such Pareto fronts have been obtained varying some of the parameters used for the optimization cycle, directly exploiting the capabilities available from the multidisciplinary tool. Analogous surveys can also be defined from the available simulation blocks, considering for example the application of the same methodology to other engineering issues that can occur during the development phases of a complex system.
These results showed how and with which advantages a model based infrastructure can be linked with analysis environments, allowing also to better understand which aspects can be better defined and formalized. The main aim of the reference case was mainly represented by the feasibility of such an approach, paving the way for other possible improvements. In this case specific implementation choices have been taken on the basis of the final purposes but other directions can also be considered. The same conceptual infrastructure can in fact be actually implemented considering other possible solutions.

Chapter 9

Critical Assessment, Further Work and Summary Conclusions

9.1 Critical assessment

The proposed methodology has been mainly developed taking into account some of the most challenging problems that characterize the design and analysis of complex systems. In particular a model-based approach has been considered and evaluated through the definition and implementation of a prototype framework. The main assumptions for the proposed infrastructure are based on the fact that the overall process can rely on already validated analysis models and tools that are the de facto standard for a specific discipline. The model-based approach available from the current work basically reflects the need for a system engineering infrastructure that is able to consistently manage multiple sources of information across different domains. Activities of optimization processes or trade-off surveys can be improved through a more structured organization. The available solutions are not well defined and mature enough to manage all the possible situations that can be faced during the development of a complex system. The already developed tools for System Engineering provides often useful capabilities that are, however, strictly related to the specific features of the tools themselves.
The proposed approach highlighted interesting results with respect to the actual methodology for the management of system information. A model-based methodology for options/alternatives management has been formulated but such solution is not the only one. Other alternatives can also be evaluated starting from the definition of different conceptual infrastructure and including for example some other aspects that have been neglected or not properly considered. The high-level concepts have been developed keeping in mind the main processes and procedures that characterize the design and verification of aerospace systems. Slightly different meta-models can arise considering the main features of other industry domains also if some of the considered formalizations are common to other fields. The connection between analyses and modeling environments, with particular emphasis from the system level perspective, represents a first attempt for the development and implementation of a model-based methodology for the enhancement of system performances. The infrastructure showed the capability to improve the current design phases but some aspects can be further enhanced through a better definition and detailing of the already developed objects and classes. Some of the features that are directly related to the connection with the modeling framework must be better detailed, ensuring a consistent representation of system data and avoiding for example data duplication when not necessary. The developed infrastructure has been implemented considering also a deeper integration with domain-specific tools, allowing for example the direct connection with the models developed in other environments. At the current status such possibility has not yet been evaluated also because such integration must take into account the development of the correct web-based services for the data sharing among the models.

9.1.1 Contributions and benefits

The current work demonstrated the benefits that can be achieved through a model-based approach for the management of system data. In particular the connection with simulation environments as well as the formalization of design options highlighted some interesting results. The correct definition of the quantities and the objects related to system development allows to speed up the design process. The reference case has been managed through the main features of the proposed architecture and such example showed how trade-off analyses can be formally defined and better exploited. In this case the conceptual relationships among system alternatives/options as well as the creation of design variables has paved the way for the correct set up of optimization and sensitivity surveys. In this way they can be represented in a more effective manner, avoiding all the problems that generally arise from information sharing and enhancing the error-prone process of data exchange.
The implementation of the proposed infrastructure on web-based platform represents one of the most promising solution for the actual realization of collaborative environments. This aspect is especially important when the modeled systems involve a large amount of resources and persons with different backgrounds and often working with different tools. The use of a web-based platforms can clearly speed up the training process of the users thanks to the wide spread of such technology among people. In this context the costs related to the training activities are basically lower with respect the introduction of dedicated desktop applications.
The advantages that can be seen from the reference case are mainly related to the straightforward capability to manage system level trade-off within a collaborative environment, allowing the people working on the same project to access a common base for data exchange. The benefits that can be achieved through such an effective way of information management are particularly evident if compared with traditional approaches for trade-off analyses, that are often strictly affected by company know-how or users' experiences. A formalized way of data processing basically results in a reduction of time and costs, ensuring also a better knowledge sharing and resource exploitation. The options management at system level can be organized in a more structured way that allows to better trace the whole information along the overall design process of the product and across multiple domains.
The application of the proposed environment to the reference case has given us the opportunity to better understand the concepts that need to be introduced or modified to model scenarios mistakenly not taken into account. In particular the current work has highlighted how some aspects, related to the integration with analysis environments, can be improved. The development of optimization capabilities within the context of a model-based framework represents one of the targets directly visible from the MBSE roadmap. The issues encountered during the implementation phases as well as the solutions taken during this survey can help to figure out what processes and features can be improved, paving the way for future enhancement of MBSE methodology.
Another important benefits highlighted by the proposed infrastructure is represented by the possibility to extend the same approach also to other fields in addition to the one directly linked to space applications (mainly considered in this study). The same concepts elaborated in the meta-model can in fact be applied to the other domains thanks to the high level representation of a complex system. Unmanned Aerial Vehicles as well as Biomedical applications/systems can in fact be approached in the same way, exploiting all the capabilities available within the platform itself.

9.1.2 Drawbacks

The proposed infrastructure has been conceived to manage project information (above all from system level perspective) through the implementation of a web-based platform. This approach must be carefully managed to avoid leak of information and problems related to users access.
Data access and information sharing must in fact be properly regulated to avoid the exchange of sensitive information. In the current work the management of such aspect is not fully considered since the purpose of the proposed approach is not mainly addressed towards this problem. The implemented framework basically exploits the control functionalities available from the native libraries for management of data

accesses. Other solutions with respect to such features can also be considered but is not the primary field of investigation of the current work.
Initiatives working on the same topics are currently evaluating the use of model-based framework through already developed tools and languages (as SysML for example) and a large amount of efforts is needed to proper integrate analysis environments. The correct formalization of the involved concepts covers a key-role for the definition and implementation of model-based infrastructures and this phases often requires the allocation of a wide set of resources. This process may seem not so important while it is fundamental to clearly pursue a well structured meta-model to avoid issues and misunderstandings when the overall framework is used operatively. This activity is generally not reflected with the same characteristics in the traditional approaches where the processes, the tools and the people are basically linked on the basis of specific knowledge and experiences (which are often the result of the solutions to similar problems gained during the years for a particular company). In these cases the main attention focuses on the operative capability to face problems as soon as possible, neglecting the fact that a structured organization of the data provides benefits on long-term projects. For this reason the evaluated approach needs a certain re-thinking of the actual processes of design, requiring in particular additional time to spent on such new methodologies with respect to already well-founded methods and practices. From this perspective such phase can be initially seen as a drawback on the way of application of such innovative philosophy.
Some of the developed concepts are also not completely validated since some of the related functions need to be widely assessed to verify their correctness and such phase requires an extensive use of the proposed approach on actual scenarios. For this reason the proposed infrastructure can be affected by further elaborations that can allow to better manage a more complete set of possible situations. This process can be done through a deeper investigation of the concepts available from the main meta-model and with a set of trials on the developed platform. Only in this way it is possible to figure out what can already be modeled and what is not yet expected.

9.2 Further work

The present work has highlighted how some interesting topics can be further analyzed and studied to improve the features related to the proposed approach and framework. Starting from the developed web services some utilities can be added, increasing the design capabilities offered. The conceptual data model used results both from current ESA standard and company expertise about collaborative engineering. The main aim of the present work has been also represented by the investigation of MBSE methodologies within the context of an aerospace modeling and analysis process. Further developments can be obtained from the integration of the current considered data model and other experience for similar projects. The identification and formalization of a common data format and a shared data structure for the exchanged information play a key role for the spreading of such MBSE methodologies in the near future.

9.2.1 Ongoing features

One of the on-going features that is now under evaluation for possible future development regards mainly the definition of a series of other web-based services for the management of analysis resources. Currently the simulation items rely on computational resources directly bounded to the main application server machine. The values required by the single simulation item could be provided in fact not by a directly integrated simulation but by properly developed web-based services. In particular a web service can be properly developed for a specific class of analysis models through which the related resources are also managed and made operative. The main system modeling framework, where the survey features are defined, can communicate with such web services when some simulation results or other information are required. This approach could contribute to increase the robustness of the overall network infrastructure, since all the information are not necessarily stored in the same place, paving the way also to the spreading of a more collaborative and distributed workspace.
One of the concepts that is already under refinement for the integration within the overall data structure

concerns the modeling of initial state and conditions with respect to a particular simulation case. Such definitions are directly related with the concepts of *Simulation Case* through the *Scenario*. The current formulation considers the distinction between the concept of *Initial State* that identify the overall set of initial values for a specific *Scenario* and the *Initial Condition* that instead is associated with the individual *VpValue*. A specific *Scenario* can be associated to zero or only one *Initial State* since if a *Scenario* needs to define new *Initial State* that more properly means that a new *Scenario* must be implemented. As defined the *Initial State* collects more generally a set of initial values (one or more) that can be identified with the term *Initial Condition*. This last class can also be named *Initial Value* that maybe better reflects the fact that it is associated to only one value of a certain property. What it is important is however the basic meaning that has been introduced with such object. The *Initial Condition* object can not only be conceived to be associated with the individual *VpValue* but also with the *Element Aspect Occurrence* class. Such association is currently under evaluation within the metamodel and it has been conceived to ensure the future possibility to model a set of equal initial conditions for a "group" of physical entities. The main idea is represented by the capability to extend the application of initial conditions not only to individual properties (related for example to lumped parameter models) but also to a set of elements (related for example to finite element models where a set of initial temperatures can deal with a large group of nodes). In this way the final purpose is to include within the metamodel the capability to capture also more complex situations when a simulation case will be defined.

9.2.2 Future developments

The large part of efforts will be addressed in the future towards two main directions. In the first one new concepts will be introduced while the current ones will be refined and enhanced to take into account for unexpected situations. The present work allowed in fact to show some situations that can be managed in a more effective way through the introduction of other objects and conceptual classes (from a meta-model perspective). Currently the conceptual infrastructure contains for example some preliminary classes for the modeling of the activities that are related to the system. These classes can be deeper formalized to proper support the definition of design activities, assembly-integration and test (AIT) activities, verification activities, operational activities, dismissal activities, etc. for example. These objects can in fact be used to organize and enhance the workflows of resources that characterize a particular product. The second research field is instead addressed to the improvements that can be achieved with a deeper integration with external solving and modeling environments.

9.2.3 Conceptual infrastructure improvements

Some of the most interesting enhancements can be pursued through a deeper investigation of the concepts related to the simulation and analysis aspects. In this case the main problem regard the capability to develop an high level structure that is able to include the large part of all the possible analysis scenarios. The large amount of possible conditions and situations that characterize external solving environments is difficult to formalize. Part of the main efforts in the current work has been used in fact to conceptualize a first pattern for such aspect but a deeper analysis can also be developed in future activities. The formalization of these concepts is not so easy to achieve since the analysis and simulation approaches used by different domains have few elements in common. Each engineering domain has its own processes, templates and analysis environments that change not only from a company/organization to another company/organization but they are often different within the same working group. The company expertise (related to the knowledge gained across the years) as well as the background of the individual user affect the development of common infrastructure. For this reason the development of new concepts and the refinement of the existing ones can help to better formalize the proposed solution.
For example in the future the *Design Variable* class may be conceptually modified to take account for the presence of nested design variables but the related formalization is currently under development to understand the actual benefits (if any) can be obtained. This situation can be useful to manage some special design situations and the related conceptual relationship can be modeled with a self-containing link for

the *Design Variable* class. This situation can be represented by the presence of *Design Variables* that are related to another *Design Variable*. That implies the possibility that the *Design Variable* can contains other Design Variables (as the self-containing relationship has highlighted). This formalization seems to be useful in the case some particular *Design Variable* depends on the definition of another Design Variable in the same *Element Definition* element. In this case both the *Design Variable* are not on the same design level but a different definition can be implemented to model it.
One of the interesting features under evaluation regards the creation of a new baseline starting from the data available on a previous one. In this way it is possible to import all that information and start from these ones to develop the new configuration and baseline. The elements can then be detailed starting from the information available as the development process proceeds. This approach shall allow to instantiate the new elements that must be distinct from the previous ones (defining new unique identifiers), avoiding the possibility that some elements may be deleted on the previous closed baseline. The elements for the new baseline must be recreated copying all the elements defined in the previous one but they must be distinguished (all the elements must be independent over all the nested levels that the system project can identify).
Some improvements can be done in the context of meta-model definitions about a deeper integration between the operative modes and scenarios with respect to the analyses and simulations conceptual classes. Future meta-model enhancements are related to a better and more compact formulation for the design variables class. For example the distinction between a discrete/continuous design variable and a group of alternatives is not so different. Both these objects have a nominal values since they are not able to represents more than one value at time. Once a belonging range has been defined (a continuous range or enumeration in the case of individual design variable and a group of alternatives) the choice among the available solutions is mutually exclusive in both cases. Also a configuration/solution that have to be chosen among a set of available ones is basically an element that can be represented as an individual design variable.

9.2.4 External environment integration

Further improvements can be achieved through the implementation of specific adapters for the definition of exchange capabilities with respect to some widespread formats. Standard exchange formats as STEP can potentially be managed through proper integrated import/export utilities within the already developed infrastructure. In particular the packages provided by some open-source initiatives can be evaluated to exploit already implemented tools to manage such kind of files.
An example of well-documented initiative is represented by the Open CASCADE project [94]. CASCADE acronym stands for Computer Aided Software for Computer Aided Design and Engineering while the overall infrastructure is often identified with the acronym OCCT that stands instead for Open CASCADE Technology. OCCT is a powerful software platform for development of CAD, CAM and CAE applications. It features 3D modeling kernel consisting of reusable C++ object libraries, and a set of development tools, all available in Open Source. It includes C++ elements for 3D surface and solid modeling, visualization, data exchange and rapid application development. The typical resources built with the support of OCCT are pre- and post-processors for finite element analysis software, CNC/CMM path generators, numerical simulation programs, etc. Open CASCADE is based on a modular structure which is conceptually represented in figure 9.1.

Data exchange is a key element in using OCCT (as well as the applications based on it) concurrently with other software such as CAD systems (PLM platforms, etc.). The standardized data exchange ensures the openness of the Open CASCADE Technology in a multi-software infrastructure, allowing it to process external data and providing a well structured level of integration at the same time. Exchanges are realized through standards which can be used between various software packages for CAD, PDM, etc. and are basically focused on IGES and STEP formats. Other connector types can work instead on proprietary formats or call run-time libraries to process external data, providing in this way the capability to interface various environments.
The reading and writing functions of 3D data as IGES format (5.3) and STEP format (AP203, AP214 and

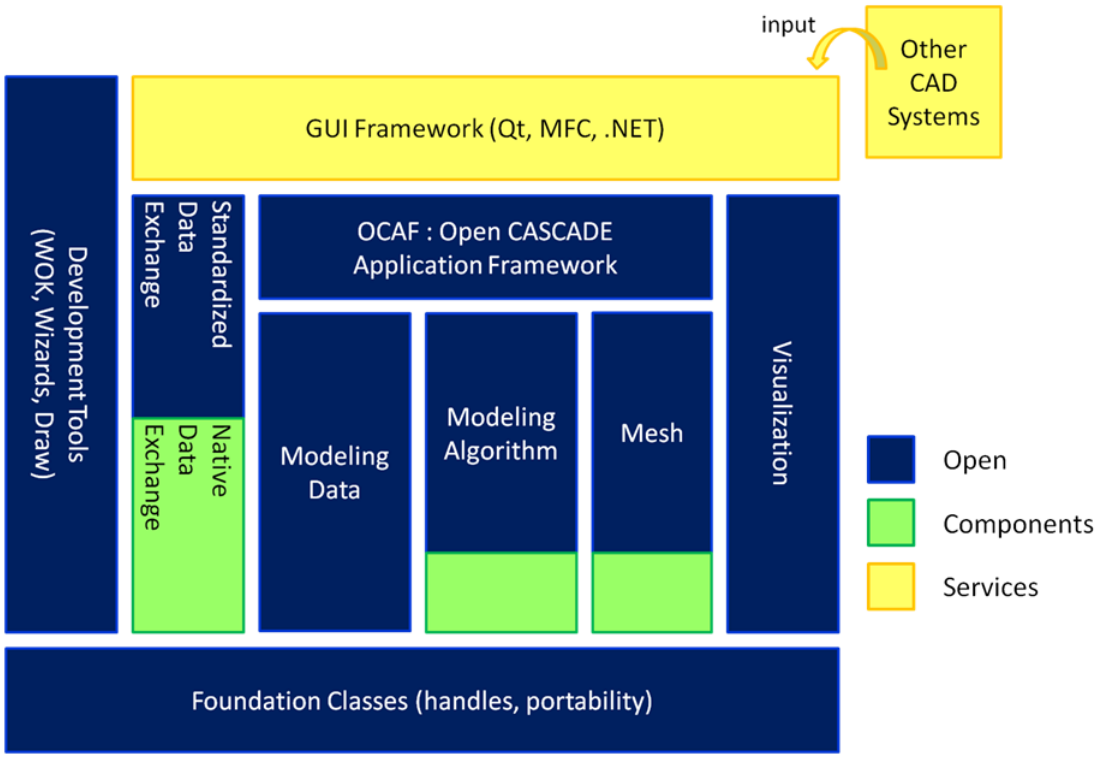

Figure 9.1: Modular structure of Open CASCADE platform [94].

AP209) are mainly used for the management of the following information:

- 3D geometry and topology.
- with Extended Data Exchange (XDE) module:
  - Colors and Names.
  - Assembly structures
  - Layers
  - Validation Properties

Specific modules are available for data analysis, adaptation, quality upgrading as well as the "customization" of shapes, regarding mathematical definitions of geometry and topology.
The main modules concerning the modeling data activity are represented by the libraries:

- 2D Geometry
- 3D Geometry
- Geometry Utilities
- Topology

Each one includes useful functionalities that can be used for different purposes: objects parametrization, data conversion, interpolation of a set of points, approximation of curves and surfaces from points, direct construction of algorithms, etc. The topology library is used for example to build pure topological data structures, defining the relationships between simple geometric entities.
The capability to parse and process STEP format is one of the most interesting one for the integration with the proposed infrastructure of the current work. Other already implemented solutions can also be considered with respect to such features. The offered functionalities can be hard coded within the developed platform and directly linked to the already defined system components and properties, allowing a clearer

handling of the information gathered and shared. In this way product data coming from different industry partners or suppliers working on the same project can be shared through STEP and managed independently on the basis of specific company platforms, tools and procedures. Such functionalities can then be used to convert the models elaborated within domain specific environments and exported through STEP format. Such files can in fact be mapped to the data structure available for the system model and the information can be collected within the *Product Model* object, providing useful instruments for the comparison with the current status of the design baseline. Data stored in the STEP file can in fact be processed to be translated in the corresponding objects of the *Product Model* tree, saving values and all the information needed exploiting the concept of *Product Element Occurrence*.

9.3 Summary conclusions

The main aim of the present work has been addressed towards the investigation of the current open issues and actual benefits that characterize the application of Model Based System Engineering methodologies in the advanced phases of a design process. In particular the options and alternatives management process has been studied through the use of a model based approach, proposing one of the possible solutions for such aspect. The integration of Multidisciplinary Analysis techniques has also been considered in the same context, highlighting the advantages that can be achieved. The application of a model based infrastructure show how the data exchange, collaboration and information consistency can be improved, paving the way for an effective support of design activities. The proposed infrastructure as well as the integration of the multidisciplinary design methods have mainly been used to assess the actual status, the current implementations and future improvements of a promising model based methodology. Such field is currently one of the most challenging research topic since it directly involves a wide set of possible improvements in the context of complex systems design and analysis. The actual implementation has been used only for the sake of clarity with respect to the investigated approach with respect to simulations and multidisciplinary analyses for example. The same infrastructure can in fact be actually implemented with other solutions or technologies but the main focus of the current study is represented by the correctness of the developed concepts.
One interesting feature related to the implementation of a web-based modeling tool is represented by the possibility to manage also simulation execution and results. The system modeling tool does not directly manage all the simulation models but support the system characteristics definition. The developed infrastructure from this point of view has been conceived in fact to properly set up simulations and analyses while solving capabilities are allocated on external analysis environments. In particular such frameworks provide also the modeling capabilities needed to define product features and its behavior.

Bibliography

[1] "A practical guide to SysML, the System modeling language", Sanford Friedenthal, Alan Moore, Rick Steiner, The MK/OMG Press.

[2] "Managing the Development of Large Software Systems", Royce, Winston W., Proceedings of IEEE WESCON 26, pp. 1-9, Aug. 1970.

[3] "A Spiral Model of Software Development and Enhancement", Boehm, Barry W., Computer, pp. 61-72, May 1988.

[4] "The Relationship of Systems Engineering to the Project Cycle", Forsberg, Kevin and Harold Mooz, Engineering Management Journal, 4, No. 3, pp. 36-43, 1992.

[5] "Simulation Modeling and Analysis", Fourth Edition, Averill M. Law, McGraw Hill.

[6] "Model Building in Mathematical Programming", H. Paul Williams, Fourth edition, Wiley.

[7] "NASA System Engineering Handbook", SP-2007-6105 Rev 1 Final 31 Dec 2007.

[8] "Survey of Model-Based Systems Engineering (MBSE) Methodologies", Jeff A. Estefan, Jet Propulsion Laboratory, INCOSE MBSE Focus Group.

[9] http://www.ecss.nl

[10] "International Council on Systems Engineering (INCOSE), System Engineering Vision 2020", Version 2.03, TP-2004004-02, September 2007.

[11] "MBSE for European Space-Systems Development", H. Eisenmann, J. Miro, H. P. De Koning, INCOSE Insight, December 2009.

[12] "Systems Engineering Guidebook: A Process for Developing Systems and Products", Martin, James N., CRC Press, Inc.: Boca Raton, FL, 1996.

[13] "Service Orient or Be Doomed!", Bloomberg, Jason and Ronald Schmelzer, John Wiley & Sons: Hoboken, New Jersey, 2006.

[14] "INCOSE MBSE Initiative Summary", Sanford Friedenthal, NDIA M&S Committee, June 15, 2010.

[15] "Space Mission Analysis and Design, 3rd edition", Wiley J. Larson, James R. Wertz, Space Technology Library, Vol. 8.

[16] "Semantically-Rigourous System Engineering Using SysML and OWL", Steven Jenkins, Nicolas Rouquette. Jet Propulsion Laboratory. SECESA 2012.

[17] "NAFEMS - INCOSE Collaboration Kick Off", International Workshop 26-29 January 2013 Jacksonville, FL, USA.

[18] http://www.esa.int/SPECIALS/CDF.html

[19] "ESA Open Concurrent Design Server", M. Bandecchi, A. Matthyssen, 2nd Concurrent Engineering for Space Applications Workshop 2006, ESA ESTEC, Noordwijk, The Netherlands, 19 – 20 October 2006.

[20] "Responsive Space System Engineering: methodologies and tools prototype", Luca Simonini. SECESA 2012 - Alameda Campus of IST / Technical University of Lisbon, Lisbon, Portugal.

[21] http://www.vsd-project.org.

[22] "ESA Virtual Spacecraft Design, Demonstration of Feasibility of MBSE Approach for European Space Programs", Harald Eisenmann, Joachim Fuchs, Don de Wilde, Valter Basso, 5th International Workshop on Systems & Concurrent Engineering for Space Applications, SECESA 2012, Lisboa, October 2012.

[23] "Multi-disciplinary Approach for Industrial Phases in Space Projects, Evolution of Classic SE into MBSE", Harald Eisenmann, Joachim Fuchs, INCOSE IW 2012, MBSE Workshop, 21-22 January 2012.

[24] "Concurrent engineering approach to design mission feasibility studies at CNES", JL. Le Gal, Collaboration and Interoperability Congress, Colorado Springs, May 21-23, 2013.

[25] "Concurrent Engineering Meta Data-Model & Multi-Domain Representation", SECESA 2012, Alameda Campus of IST/Technical University of Lisbon, Lisbon, Portugal, 17-19 October.

[26] "Applying Collaborative System Engineering in Thales Alenia Space: lessons learned and best practices", Fabio Di Giorgio, Valentina Paparo, Valter Basso, Xavier Roser, 5th International Workshop on Systems&Concurrent Engineering for Space Applications SECESA 2012, 17-19 October 2012, Lisbon, Portugal.

[27] "Dynamic Gate Product and Artifact Generation from System Models", Maddalena Jackson, Cristopher Delp, Duane Bindschadler, Marc Sarrel, Ryan Wollaeger, Doris Lam. Jet Propulsion Laboratory. Pasadena.

[28] "Computer as Thinker/Doer: Problem-Solving Environments for Computational Science", Stratis Gallopoulos, Elias Houstis and John Rice, IEEE Computational Science and Engineering.

[29] "A web-disctributed problem-solving environment for engineering applications", Hsien-Chie Cheng, Chiu-Shia Fen, Advances in Engineering Software 37, Elsevier.

[30] "A Problem Solving Environment Portal for Multidisciplinary Design Optimization", Ju-Hwan Kim, Ho-Jun Lee, Sang-Ho Kim, Jeong-Oog Lee, Advances in Engineering Software 40, Elsevier.

[31] "Problem solving environments in aerospace design", A. J. Keane, P. B. Nair, Advances in Engineering Software 32, Elsevier.

[32] "Approaches to Multidisciplinary Design Optimization", Timothy W. Simpson, Acknowledge support from the Office of Naval Research under ASSERT Grant N00014-98-1-0525.

[33] "A Unied Description of MDO Architectures". Andrew B. Lambe and Joaquim R. R. A. Martins University of Toronto, Toronto, Canada, lambe@utias.utoronto.ca. University of Michigan, Ann Arbor, Michigan, USA, jrram@umich.edu. 9th World Congress on Structural and Multidisciplinary Optimization June 13 - 17, 2011, Shizuoka, Japan.

[34] "Reconfigurability in MDO problems synthesis", part 1 and part 2. N. M. Alexandrov, R. M. Lewis. In Proceedings of the 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference 2004.

[35] "Multidisciplinary design optimization: A Survey of architectures" J. R. R. A. Martins and A. B. Lambe, AIAA Journal, 2013.

[36] "Design Structure Matrix Methods and Application", Steven D. Eppinger, Tyson R. Browning. Engineering System MIT Press.

[37] "Design Structure Matrix Methods and Applications", Steven D. Eppinger and Tyson R. Browning, Engineering Systems, MIT Press.

[38] "Network Design Using Hierarchical Performance Models and Multi-Criteria Optimization", Mingyan Liu, John S. Barasand, Center for Satellite and Hybrid Communication Networks, University of Maryland, http://www.isr.umd.edu.

[39] www.vrand.com/VisualDOC.html.

[40] http://www.esteco.com/modefrontier.

[41] http://ichrome.eu/nexus/overview.

[42] "A distributed computing environment for multidisciplinary design", Weston RP, Towsend JC, Eidson TM, Gates RL, 5th AIAA/NASA/ISS MO Symposium on Multidisicplinary Analysis and Optimization, AIAA-94-4372-CP; September 1994, p.1091-7.

[43] "IMAGE: tutorial, Version 1.17", Aerospace system design lab., Georgia Institute of Technology; 1999.

[44] "The Development of an Open-Source Framework for Multidisciplinary Analysis and Optimization", K. T. Moore, B. A. Naylor, and J. S. Gray, in 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Victoria, Canada, 2008.

[45] http://openmdao.org.

[46] "AMPL: A Modeling Language for Mathematical Programming, 2nd ed", R. Fourer, D. M. Gay, and B.W. Kernighan. Duxbury Press/Brooks/Cole Publishing Co., Pacific Grove, CA, 2003. For small examples, e.g., at most 300 variables, a student version of AMPL suffices; see http://www.ampl.com/DOWNLOADS.

[47] "Python in a Nutshell", A. Martelli. O'Reilly and Associates, Cambridge, MA.

[48] "Automated Sensitivity Analysis in Early Space Mission Design", Volker Schaus. SECESA 2012 - Alameda Campus of IST / Technical University of Lisbon, Lisbon, Portugal.

[49] "Analyses Made to Order: Using Transformation to Rapidly Configure a Multidisciplinary Environment", Cole B., IEEE Aerospace Conference; 5-12 March 2011; Big Sky, MT; United States.

[50] "Early Formulation of Model-Centric Engineering on NASA's Europa Mission Concept Study", Todd Bayer, Seung Chung, Bjorn Cole, Brian Cooke, Frank Dekens, Chris Delp, I. Gontijo, Kai Lewis, Mehrdad Moshir, Robert Rasmussen, David Wagner, Jet Propulsion Laboratory, California Institute of Technology.

[51] IEEE1471-2000(ISO/IEC 42010).

[52] "Model Based Systems Engineering (MBSE) Applied to Radio Aurora Explorer (RAX) Cubesat Mission Operational Scenarios", Sara C. Spangelo, James Cutler, Louise Anderson, Bjorn Cole et al., University of Michigan, California Institute of Technology, Jet Propulsion Laboratory, InterCAX, Exton, Phoenix Integration. IEEEAC paper2170, 2013.

[53] "Multidisciplinary Design Optimization for Concurrent Engineering of Space Systems", Jian Guo, Luca Guadagni. SECESA 2012, Lisbon, 17-19 Oct 2012.

[54] "The General Mission Analysis Tool (GMAT): A New Resource for Supporting Debris Orbit Determination, Tracking and Analysis", Moriba Jah, Steven Hughes, Matthew Wilkins, Tom Kelecy, AFRL PA 377ABW-2009-0295.

[55] "DARTS Lab, Spacecraft Modeling and Simulation", Abhinandan Jain, Jet Propulsion Laboratory.

[56] "ECSS System Glossary of terms", ECSS-S-ST-00-01C, 1 October 2012, www.ecss.nl.

[57] "ESA Virtual Spacecraft Design, Demonstration of Feasibility of MBSE Approach for European Space Programs", Harald Eisenmann, Joachim Fuchs, Don de Wilde, Valter Basso, 5th International Workshop on Systems & Concurrent Engineering for Space Applications, SECESA 2012, Lisboa, October 2012.

[58] "Human Spaceflight Mission Analysis and Design", Wiley J. Larson, Linda K. Pranke, John Connolly, Robert Giffen. Space Technologies Series, Mc Graw Hill.

[59] "NASA QUDT Handbook, Ontology-based Specification of Quantities, Units, Dimensions and Types", Ralph hodgson, Jack Spivak, The 15th NASA-ESA Workshop on Product Data Exchange, Colorado Springs, USA, 21-23 May 2013.

[60] "STEP for Data Management, Exchange and Sharing", Julian Fowler, British Library.

[61] ECSS-E-TM-10-25A 20 October 2010.

[62] ECSS-E-TM-10-21A 16 April 2010.

[63] ECSS-E-ST-7032.

[64] "ISO-15288, OOSEM and Model-based submarine Design", Paul Pearce, Matthew Hause, SETE AP-COSE 2012.

[65] "OCDT Software Design Document", OCDT-003-SDD. H.P. de Koning, P. Pinto. 2012-10-15.

[66] "Object Oriented Systems Engineering", S. Friedenthal Process Integration for 2000 and Beyond: Systems Engineering and Software Symposium, New Orleans, LA, Lockheed Martin Corporation, 1998.

[67] "Model Driven Engineering and Ontology Development, 2nd ed.", D. Gasevic, D. Djuric, V. Devedzic , Springer-Verlag.

[68] "A Research Roadmap towards Achieving Scalability in Model Driven Engineering", D. S. Kolosos, L. M. Rose, N. Matragkas, R. F. Paige et al., BigMDE 2013, Juine 2013 Budapest, Hungary.

[69] "Applying Model Based Systems Engineering (MBSE) to a Standard CubeSat", Sara Spangelo, David Kaslow, Chris Delp, Elyse Fosse, Brett Sam Gilbert, Leo Hartman, Theodore Kahn, James Cutler, 978-1-4577-0557-1/12 © 2012 IEEE.

[70] "JPL community View on Challenges and Rewards of MBSE", Bjorn Cole , Jet Propulsion Laboratory. International Workshop 26-29 Jan 2013 Jacksonville, FL, USA.

[71] "Genetic algorithms for navigating expensive and complex design spaces", D. C. Zimmerman., September 1996. Final Report for Sandia National Laboratories contract AO-7736 CA 02. 151, 158.

[72] "rSPQ++: An Object-Oriented Framework for Successive Quadratic Programming", Roscoe A. Bartlett, Lorenz T. Biegler. Department of Chemical Engineering, Carnegie Mellon University.

[73] "An Exploration of Alternative Approaches to the Representation of Uncertainty in Model Predictions", Helton, J.C., Johnson, J.D. and W.L. Oberkampf. Reliability Engineering and System Safety Vol. 85, pp. 39-71, 2004.

[74] "A sampling-based computational strategy for the representation of epistemic uncertainty in model predictions with evidence theory", Helton, J.C., Johnson, J.D., Oberkampf, W.L. and C.B. Storlie. Sandia National Laboratories Technical Report SAND2006-5557.

[75] "Epistemic Uncertainty Quantification Tutorial", Laura P. Swiler, Thomas L. Paez, Randall L. Mayes, Sandia National Laboratories, New Mexico.

[76] "Multifidelity Modeling for Uncertainty Quantification and Optimization in Design of Complex Systems", Karen Willcox, Doug Allaire, Andrew March, Leo Ng. 7th Research Consortium for Multidisciplinary System Design Workshop Purdue University July 19, 2012.

[77] "DAKOTA, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 5.0 User's Manual" Sandia Technical Report SAND2010-2183, December 2009. Adams, B.M., Bohnhoff, W.J., Dalbey, K.R., Eddy, J.P., Eldred, M.S., Gay, D.M., Haskell, K., Hough, P.D., and Swiler, L.P. Updated December 2010 (Version 5.1) Updated November 2011 (Version 5.2).

[78] "OpenMDAO Development and Usage, What's New in OpenMDAO", Kenneth T. Moore July 19th, 2012.

[79] "OpenMDAO: Framework for Flexible Multidisciplinary Design, Analysis and Optimization Methods", Christopher M. Heath and Justin S. Gray, NASA Glenn Research Center, Cleveland, OH, 44135. American Institute of Aeronautics and Astronautics.

[80] "Geometric Programming for Conceptual Aircraft Design Optimization", Woody Hoburg and Pieter Abbeel Electrical Engineering and Computer Science Department University of California, Berkeley Joint work with Laurent El Ghaoui, Alex Bayen, and Andrew Packard. 7th Research Consortium for Multidisciplinary System Design Workshop. July 20, 2012.

[81] "OpenMDAO: An Open Source Framework for Multidisciplinary Analysis and Optimization", Justin Gray MDAO Branch, NASA Glenn Research Center, Cleveland, OH Kenneth T. Moorey and Bret A. Naylorz DB Consulting Group, Inc., Cleveland, OH. American Institute of Aeronautics and Astronautics.

[82] "Extensions to the Design Structure Matrix for the Description of Multidisciplinary Design, Analysis, and Optimization Processes", Andrew B. Lambe Joaquim R. R. A. Martins.

[83] "Model Based Systems Engineering (MBSE) Process Using SysML for Architecture Design, Simulation and Visualization", Gundars Osvalds, Northrop Grumman , October 20, 2011.

[84] "Ruby on Rails Tutorial: Learn web Development With Rails, Second Edition", Michael Hartl, Addison-Wesley Professional Ruby Series.

[85] "THE RAILS 3 WAY", Obie Fernandez, Durran Jordan, Jon Larkowski, Xavier Noria, Tim Pope. Addison-Wesley Professional Ruby Series.

[86] "Design of a Model Execution Framework: Repetitive Object-Oriented Simulation Environment (ROSE)", Justin S. Gray Jeffery L Briggs. 44th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit 21 - 23 July 2008, Hartford, CT.

[87] "OMG SysML version 1.1., OMG, November 2008".

[88] "Advancing the Practise of Systems Engineering at JPL", P. A. "Trisha" Jansma and Ross M. Jones, Systems Engineering Advancement (SEA) Project, Jet Propulsion Laboratory (JPL), IEEEAC paper 1031, Version 6, January 13,2006.

[89] "Service-oriented Architecture: Concepts, Technology, and Design", Thomas Erl. Pearson Education, 2005.

[90] "Study of Thermal Analysis and Design Process in MBSE Environment", D. Riposati", Politecnico di Torino, Master Degree, Luglio 2009.

[91] "ECSS Secretariat, Requirements & Standard Division", ECSS-E-TM-10-21A, Space Engineering System Modelling and Simulation. Noordwijk, The Nederlands: ESA-ESTEC, 2010.

[92] Draft ECSS-E-00A - Space Engineering - Policy and Principle.

[93] "Integrated Approach To Optimizing Small Spacecraft Vehicles And Operations", Sara C. Spangelo, James W. Cutler, University of Michigan, IAC-11-D9.2.8.

[94] www.opencascade.org

[95] "First ARTEMIS Spacecraft Successfully Enters Lunar Orbit", Fox, Karen C. , The Sun-Earth Connection: Heliophysics. NASA.

[96] "Exploration Gateway Platform hosting Reusable Lunar Lander proposed", Bergin, Chris (December 2011). NASA Spaceflight.com. Retrieved 2011-12-05.

[97] http://www.nasaspaceflight.com/2011/12/exploration-gateway-platform-hosting-reusable-lunar-lander-proposed/

[98] "Dakota, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis", Version 5.4 User's Manual, SAND2010-2183 Unlimited Release.

[99] "Elements of Structural Optimization", R. T. Haftka and Z. Gurdal. Kluwer, Boston, 1992.

[100] "Genetic Algorithms in Search, Optimization, and Machine Learning", D. E. Goldberg. Addison-Wessley Publishing Co., Inc., Reading, MA, 1989.

[101] "Dakota, a multilevel parallel objectoriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: Version 5.4 reference manual", B. M. Adams, L. E. Bauman, W. J. Bohnhoff, K. R. Dalbey, J. P. Eddy, M. S. Ebeida, M. S. Eldred, P. D. Hough, K. T. Hu, J. D. Jakeman, L. P. Swiler, and D. M. Vigil. Technical Report SAND2010-2184, Sandia National Laboratories, Albuquerque, NM, Updated Jan. 2013.

[102] http://www.omgsysml.org.

[103] "SIM: Collaborative Model-Based System Engineering Workspace for Next-Generation Complex Systems", M. Bajaj, D. Zwemer, R. Peak, A. Phung,. A. Scott, M. Wilson, 2011 IEEE Aerospace Conference Proceedings.

[104] "FUSED: A Tool Integration Framework for Collaborative System Engineering", Mark Boddy, Martin Michalowski, August Schwerdfeger, Hazel Shackleton, and Steve Vestal. 2nd Workshop on Analytic Virtual Integration of Cyber-Physical Systems (AVICPS-11).

[105] "FUSED Framework for System Engineering Hands-on Tutorial", SAE AADL 19 April 2012, Steve Vestal, Adventium Labs.

[106] "COMPASS: Component-based Architectures for Systems Synthesis", John S. Baras, Institute for Systems Research, Department of Electrical and Computer Engineering, Fischell Department of Bioengineering Applied Mathematics, Statistics and Scientific Computation Program, University of Maryland College Park. 2012 MODPROD, February 8, 2012. Linkoping University, Sweden.

[107] "Complexity Management of Space Systems through Model Based System Engineering approach", Mauro Pasquinelli, Ph.D. Thesis, March 2010.

[108] "A Multi-Code Python-Based Infrastructure for Overset CFD with Adaptive Cartesian Grids", Andrew M. Wissink, Jayanarayanan Sitaraman, Venkateswaran Sankaran, Dimitri J. Mavriplis, Thomas H. Pulliam. American Institute of Aeronautics and Astronautics.

[109] "Technical Challenges to Systems Analysis and MDAO for Advanced Subsonic Transport Aircraft", William Haller and Mark Guynn, Technical Leads for Systems Analysis and Integration Subsonic Fixed Wing Project. AIAA Aerospace Sciences Meeting January 9-12, 2012.

[110] "The Development of an Open Source Framework for Multidisciplinary Analysis & Optimization". Kenneth T. Moore and Bret A. Naylor Wyle Information Systems, Cleveland, Ohio and Justin S. Gray NASA Glenn Research Center, Cleveland, Ohio. American Institute of Aeronautics and Astronautics.

[111] "GeoMACH: Geometry-Centric MDAO of Aircraft Configurations with High Fidelity", John T. Hwang and Joaquim R. R. A. Martins University of Michigan, Ann Arbor, Michigan, 48109, United States. American Institute of Aeronautics and Astronautics.

[112] "A Multidisciplinary Optimization Framework for Control-Configuration Integration in Aircraft Conceptual Design", Ruben E. Perez and Hugh H. T. Liu University of Toronto, Toronto, ON, M3H 5T6, Canada Kamran Behdinan Ryerson University, Toronto, ON, M5B 2K3, Canada.

[113] "Multidisciplinary Design Optimization for Complex Engineered Systems: Report from a National Science Foundation Workshop", Timothy W. Simpson The Pennsylvania State University University Park, PA USA. Joaquim R. R. A. Martins University of Michigan Ann Arbor, MI USA.

[114] "A Pareto Frontier Intersection-Based Approach for Efficient Multiobjective Optimization of Competing Concept Alternatives". A Thesis Presented to The Academic Faculty by Damon A. Rousis. In Partial Fulllment of the Requirements for the Degree Doctor of Philosophy in the School of Aerospace Engineering Georgia Institute of Technology. August 2011.

[115] "Review and Unification of Methods for Computing Derivatives of Multidisciplinary Systems", Joaquim R. R. A. Martins - John T. Hwang Multidisciplinary Design Optimization Laboratory.

[116] "A New Approach to Multidisciplinary Design Optimization via Internal Decomposition", Andrew B. Lambe, University of Toronto Institute for Aerospace Studies, Toronto, ON, Canada Joaquim R. R. A. Martinsy Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI. 13th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference 13-15 September, 2010, Fort Worth, Texas, United States.

[117] "Calculation of Sensitivity Derivatives in an MDAO Framework", Kenneth T. Moore. NASA Glenn Research Center, Cleveland, OH. American Institute of Aeronautics and Astronautics.

[118] "Approaches for Engineering Design as Mixed Discrete Non-Linear Programming Problems", Bill Crossley. Includes content from previous and current graduate students Nithin Kolencherry and Satadru Roy. Research Consortium for Multidisciplinary System Design Workshop – 20 July 2012.

[119] "Virtual Construction of Space Habitats: Connecting Building Information Models (BIM) and SysML", Raul Polit-Casillas, A. Scott Howe, Jet Propulsion Laboratory, California Institute of Technology. AIAA SPACE 2013 Conference and Exposition September 10-12, 2013, San Diego, CA.

[120] "Second-Order Reliability Formulations in DAKOTA/UQ", M. S. Eldred, B.J. Bichon, 47th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference 1 - 4 May 2006, Newport, Rhode Island.

[121] "Multimodal Reliability Assessment for Complex Engineering Applications using Efficient Global Optimization", B. J. Bichon, M. S. Eldred, L. P. Swiler, S. Mahadevan, and J. M. McFarland, 48th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference 23 - 26 April 2007, Honolulu, Hawaii.

[122] "Solution-Verified Reliability Analysis and Design of Bistable MEMS Using Error Estimation and Adaptivity", Brian M. Adams, Barron J. Bichon, Brian Carnes, et al., SANDIA REPORT, SAND2006-6286, Unlimited Release Printed October 2006.

[123] "Formulations for Surrogate-Based Optimization with Data Fit, Multifidelity, and Reduced-Order Models", M. S. Eldred and D. M. Dunlavy, 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference 6 - 8 September 2006, Portsmouth, Virginia.

[124] "Multi-point Extended Reduced Order Modeling For Design Optimization and Uncertainty Analysis", G. Weickum, M.S. Eldred, and K. Maute, 47th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, 1 - 4 May 2006, Newport, Rhode Island.

[125] "Model Calibration under Uncertainty: Matching Distribution Information", Laura P. Swiler, Brian M. Adams , and Michael S. Eldred, SAND Report 2008-0632A, AIAA Paper AIAA-2008-5944.

[126] "FORMULATIONS FOR SURROGATE-BASED OPTIMIZATION UNDER UNCERTAINTY", M. S. Eldred, A. A. Giunta†, S. F. Wojtkiewicz, Jr., and T. G. Trucano, AIAA-2002-5585.

[127] "Design Under Uncertainty Employing Stochastic Expansion Methods", M. S. Eldred, C. G. Webster, P. G. Constantine, American Institute of Aeronautics and Astronautics Paper 2008–6001.

[128] "MULTILEVEL PARALLELISM FOR OPTIMIZATION ON MP COMPUTERS: THEORY AND EXPERIMENT", M. S. Eldred, W. E. Hart†, B. D. Schimel, and B. G. van Bloemen Waanders, Sandia National Laboratories, AIAA-2000-4818.

[129] "Computational Analysis and Optimization of a Chemical Vapor Deposition Reactor with Large-Scale Computing", Andrew G. Salinger, Roger P. Pawlowski, John N. Shadid, and Bart van Bloemen Waanders Sandia National Laboratoriesy, February 9, 2004.

[130] "Aircraft conceptual design for optimal environmental performance", R. P. Henderson, J. R. R. A. Martins, R. E. Perez. THE AERONAUTICAL JOURNAL JANUARY 2012 VOLUME 116 NO 1175.

[131] "A CAD-Free Approach to High-Fidelity Aerostructural Optimization", Gaetan K.W. Kenway, Graeme J. Kennedy, Joaquim R. R. A. Martinsz, 13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference September 13-15, 2010, Fort Worth, Texas, United States.

[132] "A Standard Platform for Testing and Comparison of MDAO Architectures", Justin Gray, Kenneth T. Moore, Tristan A. Hearn, Bret A. Naylorx, NASA Glenn Research Center, Cleveland, OH. American Institute of Aeronautics and Astronautics.

[133] "On Unifying Geometric Representations in an MDAO Environment with Application to Aircraft Design", John F. Dannenhoer, Robert Haimes, 7th Research Consortium for Multidisciplinary Systems Design Workshop Purdue University.

[134] "An Information-Theoretic Metric of System Complexity with Application to Engineering System Design", Douglas Allaire, Chelsea He, John Deyst, and Karen Willcox. Department of Aeronautics and Astronautics. Massachusetts Institute of Technology. 7th Research Consortium for Multidisciplinary System Design. July 20, 2012. Purdue University, West Lafayette, IN.